

(LK) initramfs

手稿

封面

本节简要介绍 Linux 内核中有关 `initramfs` 的一些内容。

概述

上一小节“Linux 的启动”中，我们谈到了 `initrd`，它或是用作常规根文件系统，或是作为中间环境，为系统挂载真正的文件系统做好准备。

2.6 内核提供了一种称为 `initramfs` 的新功能，它在几个方面较 `initrd` 更为优秀。



initrd 与 initramfs

initrd 是一种由 boot loader 加载的常驻内存的虚拟磁盘镜像，因而被称为 initramdisk 或者 initrd。由于是模拟磁盘，它会带来 Linux 块 I/O 子系统的开销。

而 initramfs 则不同，它基本上如同一个被挂载的文件系统一样，由自身获取缓冲，因此被称作 initramfs。

另外，initrd 要求内核镜像包含 initrd 所使用文件系统的驱动程序，而 initramfs 则不需要文件系统支持。

rootfs

在进一步了解 `initramfs` 之前，我们需要对 `rootfs` 有所了解。

`rootfs` 是 `ramfs`（或者 `tmpfs`，如果开启的话）的一个特殊实例，在 2.6 版本的 Linux 系统中它始终存在。如同不能杀死 `init` 进程一样，你同样不能卸载 `rootfs`。

注：`ramfs` 是一种基于内存的简单文件系统；`tmpfs` 派生自 `ramfs`，它扩充了大小限制，并且可以向交换空间写入数据。

大多数系统通常在 `rootfs` 上挂载其他文件系统并忽略 `rootfs`，空白 `ramfs` 实例占用的空间非常小。

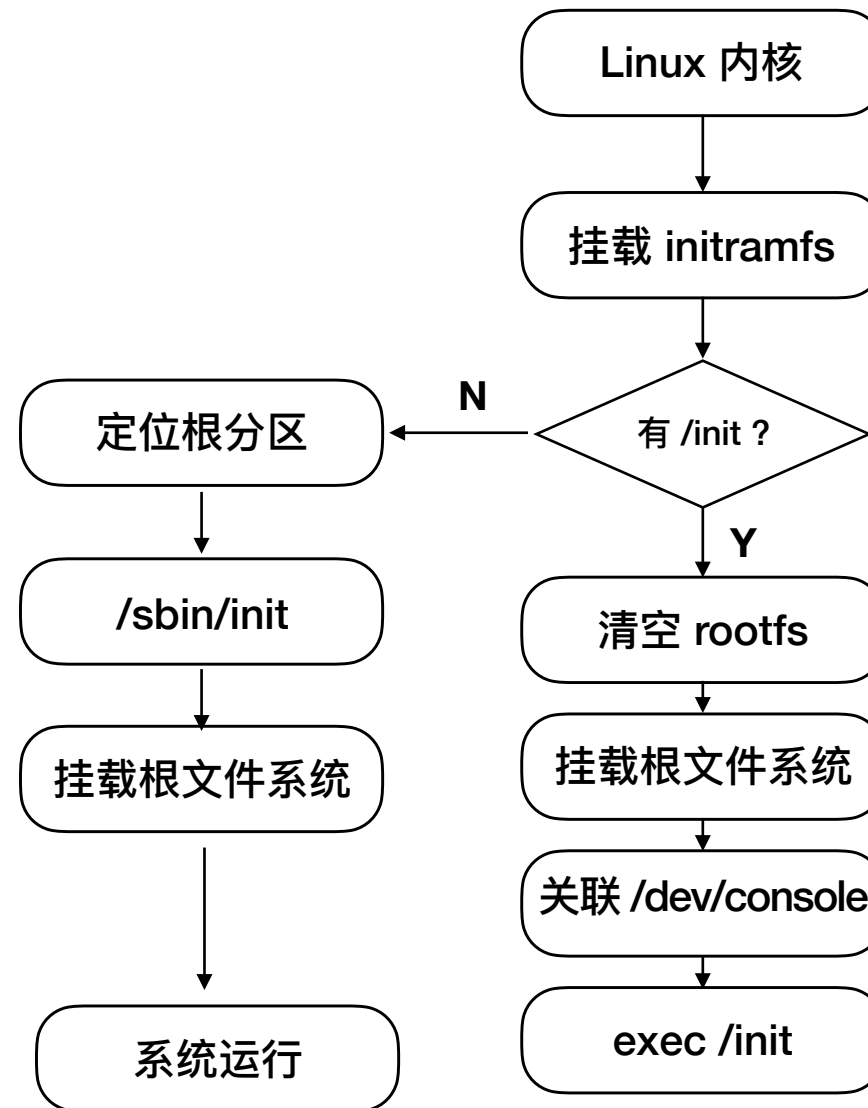
initramfs 的工作原理

initramfs 是基于页缓冲建立的，如同页缓冲一样会动态地变大或缩小，因此可以减少内存消耗。由于 initramfs 只是页缓冲之上的一小层，它的代码量很小。

所有的 2.6 Linux 内核都包含一个经过 gzip 压缩的 cpio 格式归档文件，它在内核启动时释放到 rootfs。在释放以后，内核会查看 rootfs 中是否包含一个名为 init 的文件，如果存在则执行该文件，生成进程的 PID 是 1。如果找到了，这个 init 进程负责系统启动过程的剩余部分，包括定位并挂载“真正的”根设备（如果有的话）。如果 cpio 格式归档文件在释放到 rootfs 后，里面不包含 init 程序，内核会退回到早些的代码位置，定位并挂载根分区，然后用 exec 执行类似 /sbin/init 这种变体形式的初始化程序。

启动过程（图）

Linux 使用 initramfs 时的启动流程如下图所示。



使用 initramfs 的 Linux 内核

对比 initrd

由以上介绍可知，initramfs 与旧的 initrd 相比，有如下几方面的不同：

1. initrd 总是一个单独的文件，而 initramfs 归档文件被链接到了 Linux 内核镜像。
2. initrd 文件是一个经过 gzip 压缩的文件系统镜像，而 initramfs 归档文件是经过 gzip 压缩的 cpio 归档文件。内核的 cpio 提取代码不仅极其小巧，而且是用 __init 标记的数据，它们可以在启动之后丢弃。
3. 由 initrd 运行的程序在做一些设置之后返回到内核，而 initramfs 中的 init 程序不会返回内核。
4. 当切换到其他根设备时，initrd 会 pivot_root，然后卸载 ram disk。但 initramfs 是 rootfs，你不能用 pivot_root 切换 rootfs，也不能卸载它。对于 initramfs 而言，删除 rootfs 中所有内容来释放空间，在 rootfs 基础上挂载新的根文件系统，将 stdin/stdout/stderr 关联到新的 /dev/console，然后调用 exec 执行 init。

配置 initramfs

2.6 内核构建过程总是创建一个经过 `gzip` 压缩的 `cpio` 格式的 `initramfs` 归档文件，并将它链接到生成的内核二进制文件中。默认情况下，这个归档文件是空的（x86 上消耗 134 字节）。

配置项 `CONFIG_INITRAMFS_SOURCE` 可用于指定 `initramfs` 归档文件的来源，它们会自动纳入到生成的二进制文件。

`CONFIG_INITRAMFS_SOURCE` 可以指向一个已经存在的经过 `gzip` 压缩的 `cpio` 归档文件、包含将要归档的文件的目录、或者是一个文件规范。

文件规范

用来指定 `initramfs` 的文件规范如下所示。

```
dir /dev 755 0 0
nod /dev/console 644 0 0 c 5 1
nod /dev/loop0 644 0 0 b 7 0
dir /bin 755 1000 1000
slink /bin/sh busybox 777 0 0
file /bin/busybox initramfs/busybox 755 0 0
dir /proc 755 0 0
dir /sys 755 0 0
dir /mnt 755 0 0
file /init initramfs/init.sh 755 0 0
```

构建内核之后，你可以运行 `usr/gen_init_cpio` 来获得描述以上文件格式的使用说明。

使用这种配置文件的一个优点是无需 `root` 访问权限来设置权限或者在新的归档文件中创建节点。

制作 initramfs 归档文件

好了，我们现在可以动手创建一个 initramfs 归档文件试试！

要让 initramfs 跑起来，一个比较好的开始是将 “hello world” 程序静态链接为 init，然后测试。

```
$ mkdir initramfs && cd initramfs  
$ vi hello.c
```

```
1 #include <stdio.h>  
2 #include <unistd.h>  
3  
4 int main(int argc, char *argv[])  
5 {  
6     printf("Hello initramfs!\n");  
7     sleep(999999999);  
8 }
```

```
$ gcc -static hello.c -o init  
$ echo init | cpio -o -H newc | gzip > test.cpio.gz  
1784 块
```

这个生成的 test.cpio.gz 就是我们要用来测试的 initramfs 归档文件，它只是简单地打印一个字符串 “Hello initramfs!”，然后进入休眠。

测试 initramfs

如同测试新编译的内核一样（参见“编译内核”小节），我们需要首先将 `initramfs` 归档文件拷贝到测试主机的 `/boot/` 目录下。

```
test@test:~$ sudo scp lettuce@lettuce:~/source/linux/initramfs/test.cpio.gz /boot/
```

然后修改 `grub`，在 `/etc/grub.d/40_custom` 文件末尾添加新的启动项，内核仍然使用之前编译过的内核，而 `initrd` 指定为我们制作的 `initramfs` 归档文件。

```
menuentry 'linux-4.14-test-initramfs' --class ubuntu --class gnu-linux --class gnu --class os {  
    linux    /boot/vmlinuz-4.14 ro quiet splash  
    initrd   /boot/test.cpio.gz  
}
```

最后，更新 `grub` 配置，重启虚拟机。

```
$ sudo update-grub  
$ sudo reboot
```

initramfs 的输出

当虚拟机重启后，选择新添加的启动项，这样内核就会根据 `gurb` 配置加载 `/boot/test.cpio.gz`，从而执行我们制作的 `initramfs` 归档文件中的 `init` 程序。

```
early console in extract_kernel
input_data: 0x000000000025dc276
input_len: 0x0000000000078d1f7
output: 0x00000000001000000
output_len: 0x00000000001d3af3c
kernel_total_size: 0x00000000001a25000
booted via startup_32()
Physical KASLR using RDRAND RDTSC...
Virtual KASLR using RDRAND RDTSC...

Decompressing Linux... Parsing ELF... Performing relocations... done.
Booting the kernel.
Hello initramfs!
_
```

为什么选择 `cpio`?

最后，为什么 Linux 内核选择 `cpio` 而不是 `tar` 呢？这个决定来自于 2001 年 12 月。

有关这个问题有许多讨论，下面是一些快速摘要版本：

1. `cpio` 是标准的。已经几十年了（从 AT&T 时代开始），而且广泛用于 Linux。之所以没有像 `tar` 那样流行，是因为 `cpio` 命令的参数太丑了。
2. 被内核选中的 `cpio` 归档格式较其他任何归档格式都更为简洁。
3. GNU 项目在 `tar` 上的标准几乎和 `zip` 在 Windows 上的标准化是近似的。Linux 不属于任何一方，可以自由做出技术决策。
4. 由于是内核内部格式，因此可以是全新的样子。内核提供自己的工具来任意创建和提取这种格式。使用现有技术也许更好，但不是必须的。
5. Al Viro 做的这个决定，更重要的是，设计并实现了 `initramfs` 代码。他认为 `tar` 不适合 Linux 内核，用他的话说："tar is ugly as hell and not going to be supported on the kernel side"，并且对此做出了解释。

参考资料

参考资料:

Essential Linux Device Drivers

linux/Documentation/filesystems/ramfs-rootfs-initramfs.txt

(Al Viro)

<http://www.uwsg.iu.edu/hypermail/linux/kernel/0112.2/1540.html>

<http://www.uwsg.iu.edu/hypermail/linux/kernel/0112.2/1550.html>

<http://www.uwsg.iu.edu/hypermail/linux/kernel/0112.2/1638.html>