**Initramfs的使用**

宝德技术研究院 李磊

# Summery

linux启动的时候必须启动init进程,这个用户进程实际上是调用文件系统中的一个名为init的可执行文件启动的.所以linux kernel必须先把包含init文件的文件系统挂载上.而挂载文件系统之前,需要解析根文件系统中的/etc/fstab,这就出现了”先有鸡还是先有蛋的问题”.

为了解决这个问题, 内核开发者们通过给kernel传递一个”root”这样的参数,让kernel通过这个参数找到根文件系统,并且将其挂载.但后来出现了各种各样的文件系统,比如nfs, raid等,在kernel中集成全部的文件系统是不可行的.后来就出现了initrd, 内核起来后就先挂载initrd, 后续文件系统的挂载由initrd中的init去执行.

# Ramfs的优势

块设备挂载之前需要格式化, 并且块设备的大小不能改变.Ramdisk既然是基于内存的块设备,块设备的上述问题,Ramdisk必定会有.大小固定也就会造成一部分内存的浪费,而且内存浪费不仅仅只是由大小固定造成的.在Linux中为了提高块设备的访问速度,都会在内存中开辟一块区域用于缓存,缓存的开辟无疑再次浪费了内存空间.

后来,kernel的开发者们开发出了ramfs, 这是个基于内存的文件系统,这个文件系统的大小可以根据当前使用情况自动的改变,并且文件系统这一层面上没有給块设备使用的缓存,这样就节省了内存的空间.

# Using initramfs

我们先来写一个init程序,这个程序什么都不做,仅仅打印一个”hello rocklee!”

#include <stdio.h>

int main(void)

{

printf("Hello Rocklee!\n");

sleep(999999999);

}

然后将其链接静态库,编译成一个可执行文件

**gcc -static init.c -o init**

接着我们将这个init放入一个目录,这个目录就是我们的initramfs.同时,还需要提供一个字符设备,用与输出打印, 这个字符设备的主设备号必须为5, 次设备号必须为1.在使用过程中还要保证init这个文件必须有可执行权限.

**mkdir dev**

**mknod dev/console c 5 1**

**mkdir ramfs**

**cp -rf init dev ramfs**

**cd ramfs**

**find . | cpio -o -H newc | gzip > ../initramfs\_data.cpio.gz**

**cd ..**

有如下方式可以将我们刚才做好的initramfs编译进kernel

•将initramfs\_data.cpio.gz放入kernel目录下的usr,替换掉原有的同名文件.

•编译kernel的时候在menuconfig指定CONFIG\_INITRAMFS\_SOURCE. CONFIG\_INITRAMFS\_SOURCE可以指向一个绝对路径,或者相对于kernel源码的路径.其目标可以是一个cpio.gz包,一个目录,或者是一个文本配置文件.

由于kernel在启动后,如果发现bootloader有传入initrd,就会优先使用用bootloader传入的initrd,所以在我们实验之前,先修改grub,防止grub传入的initrd覆盖内核中的initramfs.需要将grub.config中的initrd那一行注释掉.

hiddenmenu

title CentOS (2.6.18-rocklee-g0b57ceed)

root (hd0,0)

kernel /vmlinuz-2.6.18-rocklee-g0b57ceed ro root=LABEL=/ rhgb quiet

# initrd /initrd-2.6.18-rocklee-g0b57ceed.img

正常情况下,kernel启动后就会输出预期的结果.