WinnerOS文件系统定制指南

版本2017年8月第一版

WinnerOS开发团队

前言：

WinnerOS是一款模块化的操作系统，用户可自行更改自己的文件系统。也适合极客研究小型Linux的工作原理。如果你是个Linux菜鸟，又想制作全宇宙只属于你的发行版本，没有关系，我建议你可以先学习Linux的基本命令，并按照官方建议去做，你就可以拥有了，到时候你就可以装逼说我有一个限量版系统。哦不不不不，搞错了，是属于你的系统，拿去炫耀有可能没有什么亮点（除非你做的够好，甚至可以作为安装CD安装本系统）。

好了，废话少讲，开始你的WinnerOS定制吧！

1.如何将你的系统或Linux命令行小程序（如memtest）加入你的系统CD：

请按这种格式加入

LABEL os

MENU DEFAULT

MENU LABEL XXXX（你的系统名）

TEXT HELP

（写上你的帮助）

All extensions are loaded mount mode. Boot media is not removable.

ENDTEXT

KERNEL /vmlinuz（不要动）

APPEND initrd=/ramdisk所在的文件夹（小程序可以不用这行）

注意：必须按照上面的大小写来写。如果是在太懒，可以直接复制粘贴，由于配置文件不支持UTF-8，所以如果是中文说明，请用翻译软件翻译成英文。

2.如何制作你的ramdisk

我们官方推荐使用Busybox制作，当然你也可以使用其他方法，如果你只是个Linux菜鸟，建议按照官方建议来做，如因未按照官方建议来做，后果需自行承担。

可参考百度文库里的《BabyLinux制作过程详解》，以下内容均摘自于此（有改动和注释）：

3,配置busybox

busybox的配置程序和linux内核菜单配置方式简直一模一样.熟悉用make menuconfig方式配置linux内核的朋友很容易上手.

#cp busybox-1.00.tar.gz /babylinux

#cd /babylinux

#tar xvfz busybox-1.00.tar.gz

#cd busybox-1.00

#make menuconfig

下面是需要编译进busybox的功能选项,其他的可以根据需要自选,但是同样不要太贪心.

General Configuration应该选的选项

Show verbose applet usage messages

Runtime SUID/SGID configuration via /etc/busybox.conf

Build Options

Build BusyBox as a static binary (no shared libs)

这个选项是一定要选择的,这样才能把busybox编译成静态链接的可执行文件,运行时才独立于其他函数库.否则必需要其他库文件才能运行,在单一个linux内核不能使他正常工作.

Installation Options

Don't use /usr

这个选项也一定要选,否则make install 后busybox将安装在原系统的/usr下,这将覆盖掉系统原有的命令.选择这个选项后,make install后会在busybox目录下生成一个叫\_install的目录,里面有busybox和指向他的链接.

其他选项都是一些linux基本命令选项,自己需要哪些命令就编译进去,一般用默认的就可以了.

配置好后退出并保存.

STEP1——编译并安装busybox

#make

#make install

编译好后在busybox目录下生成子目录\_install,里面的内容:

drwxr-xr-x 2 root root 4096 11月 24 15:28 bin

lrwxrwxrwx 1 root root 11 11月 24 15:28 linuxrc -> bin/busybox

drwxr-xr-x 2 root root 4096 11月 24 15:28 sbin

其中可执行文件busybox在bin目录下,其他的都是指向他的符号链接.

我编译出来的busybox可执行文件是935K,加上符号链接,整个\_install目录是952K.加上845K的内核不是已经超过1440K了吗?别担心,我们将对整个根文件系统做大幅度的压缩。（由于现在软盘已经过时，不再使用，使用的是iso文件，所以不用担心空间问题。）

STEP2——制作根文件系统

1,基本目录结构

#cd /babylinux/rootfs

#mkdir etc usr var tmp proc home root dev

其中etc,proc和dev是一定要建的,bin和sbin不用建,因为busybox中已经有了.

其他的可以象征性的建几个就可以了.

拷贝busybox

#cp -R /babylinux/busybox-1.00/\_install/\* /babylinux/rootfs/

2,建立设备文件名

#cd /babylinux/rootfs/dev

你可以

用mknod手工建立,也可以直接从原系统的/dev目录下拷贝过来.

手工建立的方法:

#ls -l /dev/console

crw------- 1 root root 5, 1 11月 30 09:02 /dev/console

这样就查看到了console设备的主设备号是5,辅设备号是1,是一个标记为C的字符设备.

于是,我们可以用mknod建立一个同样的设备文件:

#mknod console c 5 1

但是手工方法建立太麻烦了,通常直接从/dev下把需要的设备文件拷贝过来.

这些设备文件是特殊文件,在拷贝时一定要加上-R参数才能拷贝.

#cp -R /dev/console ./

#cp -R /dev/null ./

#cp -R /dev/zero ./

...

以下是我认为需要的设备名:

cdrom fd0 hda14 hda4 hdb11 hdb19 hdc hdc16 hdc6 hdd13 hdd3 loop2 ram2

console fd0H1440 hda15 hda5 hdb12 hdb2 hdc1 hdc17 hdc7 hdd14 hdd4 loop3 tty0

fb hda hda16 hda6 hdb13 hdb3 hdc10 hdc18 hdc8 hdd15 hdd5 loop4 tty1

fb0 hda1 hda17 hda7 hdb14 hdb4 hdc11 hdc19 hdd hdd16 hdd6 loop5 tty2

fb1 hda10 hda18 hda8 hdb15 hdb5 hdc12 hdc2 hdd1 hdd17 hdd7 null tty3

fb2 hda11 hda19 hdb hdb16 hdb6 hdc13 hdc3 hdd10 hdd18 hdd8 ram tty4

fb3 hda12 hda2 hdb1 hdb17 hdb7 hdc14 hdc4 hdd11 hdd19 initctl ram0 tty5

fb4 hda13 hda3 hdb10 hdb18 hdb8 hdc15 hdc5 hdd12 hdd2 loop1 ram1 zero

其中,fd0,hda,ram,ram1,tty1,null,zero,loop1,fb0,fb等是必备的.

其它的hda,hda1,hdb等可以根据实际需要决定.但是上表中的选择是比较合理的,即能满足大部分的需要,有没有不用的设备浪费空间.注意,千万不要把/dev下的设备全拷贝过来,那将产生大约420K的/dev目录,这对babylinux来说太大了.（请忽略大小问题。）

3,建立etc目录下的配置文件

busybox.conf group inittab motd passwd resolv.conf shadow-

fstab init.d issue mtab profile shadow

其中init.d是一个目录,从busybox-1.00源代码目录下拷贝过来.

#cp -R /babylinux/busybox-1.00/examples/bootflopyp/etc/init.d /babylinux/rootfs/etc/

busybox.conf是一个空文件.

其他文件的内容如下:

fstab

/dev/fd0 / ext2 defaults 0 0

none /proc proc defaults 0 0

/dev/cdrom /mnt/cdrom udf,iso9660 noauto,owner,kudzu,ro 0 0

/dev/fd0 /mnt/floppy auto noauto,owner,kudzu 0 0

group

root:x:0:root

inittab

::sysinit:/etc/init.d/rcS

::askfirst:/bin/sh

tty2::respawn:/bin/getty 38400 tty2

tty3::respawn:/bin/getty 38400 tty3

tty4::respawn:/bin/getty 38400 tty4

# Stuff to do when restarting the init process

::restart:/bin/init

# Stuff to do before rebooting

::ctrlaltdel:/bin/reboot

::shutdown:/bin/umount -a -r

::shutdown:/bin/swapoff -a

issue

Baby Linux release 0.1

motd

mtab

passwd

root::0:0:root:/root:/bin/ash

profile

# /etc/profile: system-wide .profile file for the Bourne shells

echo

echo

export PS1="[\u@\h \w]\$"

echo "Done"

alias ll='ls -l'

alias du='du -h'

alias df='df -h'

alias rm='rm -i'

echo

resolv.conf

nameserver 202.96.209.5

nameserver 202.96.209.6

shadow

root:$1$$adltAB9Sr/MSKqylIvSJT/:12705:0:99999:7:::

shadow-

root:$1$DWU.tenP$B7ANiXoGoiZMwJ

R6Ih8810:12705:0:99999:7:::

其中有很多是从原系统的/etc下拷贝过来修改的,如果你是一个具有中等以上水平的linux爱好者,那么应该一看就明白了,当然,你也可以根据自己的需要修改这些文件.其中最重要的是fstab和inittab,busybox内建的init程序用到的inittab文件的语法和一般的不一样,不能直接把原系统/etc下inittab文件拷贝过来.可以把busybox-1.00目录下的示例文件拷贝过来修改用.具体请看busybox的文档.busybox的init也可以不用inittab.但是在我制作babylinux过程中有一个非常奇怪的bug.所有/sbin下的busybox链接在做成压缩的根文件系统,解压后都不能正常运行,显示找不到该命令.只有当我在/bin下做这些链接时才能运行.具体原因还不太清除,所以你需要做下面的工作:

#cd /babylinux/rootfs/sbin

#ls

chroot getty ifconfig losetup pivot\_root reboot swapoff sysctl

fdisk halt init mkswap poweroff route swapon telnetd

查看到sbin下有上述链接

转到bin下

#cd /babylinux/rootfs/bin

重新做这些链接:

#ln -s busybox chroot

#ln -s busybox getty

#ln -s busybox ifconfig

...

然后把sbin下的链接删除,以节省空间

#rm -rf /babylinux/rootfs/sbin/\*

再把原先inittab中所有的sbin改成bin

init.d下的文件:

rcS

请确保这个文件是可执行的,否则请改成可执行的:

#chmod u+x rcS

rcS的内容:

#! /bin/sh

mount -o remount,rw /

/bin/mount -a

>/etc/mtab

echo

echo

echo

echo

echo -en "\t\tWelcom to \\033[0;32mBabyLinux\\033[0;39m\n"

echo -en "\\033[0;36m\n"

echo

echo -en "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\t\t\n"

echo -en "+ This is a tiny linux system based on a floppy.It contains \t\t\n"

echo -en "+ more than 100 basic Linux commands and tools.The kernel of \t\t\n"

echo -en "+ this tiny system support all kinds of normal filesystems. \t\t\n"

echo -en "+ linux ext2,ext3,jfs,reiserfs and windows fat,vfat,ntfs[readonly]\t\t\n"

echo -en "+ is supported! So it is a powerful small system you can use it \t\t\n"

echo -en "+ as a linux and windows rescue disk.Beside this,the kernel also\t\t\n"

echo -en "+ contains the drivers of Reltek8139,NE2000,via-rhine ethernet\t\t\n"

echo -en "+ adpater. you can configure the IPaddress and netmask with tools\t\t\n"

echo -en "+ 'ifconfig' and config the default gateway with command 'route'. \t\t\n"

echo -en "+ Is there anything else? Haha,this is a telnet server build-in\t\t\n"

echo -en "+ you can type 'telnetd' to startd it and thus your friends can\t\t\n"

echo -en "+ logon to your system to help you solve the problem.\t\t\n"

echo -en "+ \\033[0;32mAll these great features are powered by BusyBox 1.0\\033[0;36m\t\t\n"

echo -en "+ This is a free system tool developed by GuCuiwen.\t\t\n"

echo -en "+ RUN YOUR OWN RISK of using it ! if you have any problem please\t\t\n"

echo -en "+ mailto : win2linux@163.com Enjoy!!\t\t\n"

echo -en "++++++

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\t\t\n"

echo -en "\\033[0;39m\n"

hostname BabyLinux

可以自己作相应的修改.

以上是babylinux根文件系统的所有内容,他的总大小应该在1M左右.

[root@gucuiwen baby]# du -hs

1.1M .

STEP3——制作ramdisk映象文件

babylinux根文件系统所有东西都放在/babylinux/rootfs下,我们将利用ramdisk把这些内容做成ramdisk映象文件并压缩他.

以下主要是ramdisk的用法,看完以下内容,你应当学会ramdisk的使用.

[root@gucuiwen babylinux]# dd if=/dev/zero of=/dev/ram1

dd: 正在写入 ‘/dev/ram1’: 设备上没有空间

读入了 8193+0 个块

输出了 8192+0 个块

zero是一个特殊的设备,表示全部为0的字符块.上面这条命令的意思是把系统的第一个ramdisk用全部为0的数据填充,因为ramdisk默认大小为4M,因此当读满8192个块(每块512字节)后,显示'设备上没空间'.这很正常,/dev/ram1已经被填充满了.

如果指定块的大小:

[root@gucuiwen babylinux]# dd if=/dev/zero of=/dev/ram1 bs=1M count=4

读入了 4+0 个块

输出了 4+0 个块

不会有错误提示,这里演示了dd的一般用法,接下来还要频繁用到dd命令.

在/dev下有很多ramdisk设备,ram1,ram2,ram3....

一般用第一个就可以了.

填充后,ram1就有可空间,可以在这个空间上创见一个文件系统:

[root@gucuiwen babylinux]# mkfs.ext2 -m0 /dev/ram1

mke2fs 1.32 (09-Nov-2002)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

1024 inodes, 4096 blocks

0 blocks (0.00%) reserved for the super user

First data block=1

1 block group

8192 blocks per group, 8192 fragments per group

1024 inodes per group

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

将ram1挂装到文件系统中:

先建立一个挂装点:

#mkdir /mnt/ram

挂上ram1:

#mount /dev/ram1 /mnt/ram

将先前做好的babylinux根文件系统拷贝到ram1上.

#cp -R /babylinux/rootfs/\* /mnt/ram

做完以上几步,你应就白了ramdisk设备的含义,他是和hda1,hdb1,一样的块设备,用mount挂到文件系统下后就可以访问,往里放东西,但是所有的东西在内存上.关机将丢失所有东西.

拷贝好babylinux根文件系统后卸载ram1:

#umount /dev/ram1

这时,虽然不能通过/mnt/ram这个挂装点访问ram1中的内容了,但是他却切切实实得在内存中存在.

再用dd把这个ram1以映象方式取出来:

[root@gucuiwen babylinux]# dd if=/dev/ram1 of=/babylinux/ramdisk.img

读入了 8192+0 个块

输出了 8192+0 个块

验证一下取出来的内容:

[root@gucuiwen babylinux]# file ramdisk.img

ramdisk.img: Linux rev 1.0 ext2 filesystem data

他是一个ext2 文件系统,类似一个ISO光盘映象文件.

因次,我们可以用loop设备来把他重新挂装到文件系统里:

[root@gucuiwen babylinux]# mount -o loop ramdisk.img /mnt/ram/

为了方便,我仍旧把他挂在/mnt/ram下,因此,在先前一定要把/dev/ram1 umount掉

查看/mnt/ram下的内容,他应该和/babylinux/rootfs下的一模一样,否则就是出错了:

[root@gucuiwen babylinux]# ls /mnt/ram

bin dev etc home lost+found mnt proc root sbin tmp usr var

这样,我们就得到了一个ramdisk根文件系统映象:ramdisk.img

把他umount掉:

#umount /mnt/ram

如果是第一次接触ramdisk,你可能对上述的内容很迷惑,如果这样,请反复阅读和理解上面的内容,自己多动手做几次试验,就可以理解.

压缩ramdisk.img印象文件:

[root@gucuiwen babylinux]# gzip -v9 ramdisk.img

ramdisk.img: 87.9% -- replaced with ramdisk.img.gz

查看压缩后的大小:

[root@gucuiwen babylinux]# ls -lh ramdisk.img.gz

-rw-r--r-- 1 root root 495K 11月 30 11:32 ramdisk.img.gz

我得到的压缩ramdisk映象文件安是495K. 加上内核的845K,是1340K

符合公式:

内核大小+文件系统压缩印象文件+50K <= 1440K

如果你做出来的kernel和ramdisk.img.gz太大了,请重新制作kernel或ramdisk.img.gz,在其中做一些取舍,如果你的kernel和ramdisk.img.gz太小了,那么可以再往里面添加一些内容,使你的babylinux功能更强.

注意：以上部分内容如果做不到可以忽略，但是划下划线的是不得不做的。Babylinux文件夹请自行建立。为保险起见，建议这些操作请在虚拟机上完成。

如有疑问：

可参考《BabyLinux制作过程详解》原文件<https://wenku.baidu.com/view/1a62da254b35eefdc8d333b6.html>

或直接发邮件：877259039@qq.com

参考资料：

《BabyLinux制作过程详解》作者:GuCuiwen <https://wenku.baidu.com/view/1a62da254b35eefdc8d333b6.html>

修正记录：

2017-8：初始版本