前提：

1、一个作为宿主机的Linux；本文使用的是Redhat Enterprise Linux 5.4；

2、在宿主机上提供一块额外的硬盘作为新系统的存储盘，为了降低复杂度，这里添加使用一块IDE接口的新硬盘；

3、Linux内核源码，busybox源码；本文使用的是目前最新版的linux-2.6.34.1和busybox-1.16.1。

说明：本文是一个step by step的实做指南；

一、为系统上的新硬盘建立分区，这里根据需要先建立一个大小为100M的主分区作为新建系统的boot分区和一个512M的分区作为目标系统（即正在构建的新系统，后面将沿用此名称）的根分区；100M的分区格式化后将其挂载至/mnt/boot目录下；512M的分区格式化后将挂载至/mnt/sysroot目录；

说明：

1、此处的boot和sysroot的挂载点目录名称尽量不要修改，尤其是boot目录，否则您必须保证后面的许多步骤都做了相应的改动；

2、新建系统的boot目录也可以跟根目录在同一个分区，这种方式比独立分区还要简单些，因此这里将不对此种方法再做出说明；

二、编译内核源代码，为新系统提供一个所需的内核（本例中的源代码包都位于/usr/src目录中）

# cd /usr/src

# tar jxvf linux-2.6.34.1.tar.bz2

# ln -sv linux-2.6.34.1 linux

# cd linux

然后下载ftp://172.16.0.1/pub/Sources/kernel/kernel-2.6.30.5-i686.cfg至当前目录中，并重命名为.config。

# make menuconfig

根据您的实际和规划选择所需要的功能；本实例计划制作一个具有网络的功能的tiny linux且不打算使用内核模块，因此，这里选择把本机对应的网卡驱动直接编译进了内核。作者使用的是vmware Workstation虚拟机，所以，所需的网上驱动是pcnet32的，其它的均可按需要进行选择。选择完成后需要保存至当前目录下.config文件中。

# make SUBDIR=arch/

# cp arch/x86/boot/bzImage /mnt/boot

# make menuconfig

提示：为了实现后面的功能，请务必将文件系统中的ext3和网卡的驱动程序直接编译进内核；

三、编译busybox

# cd /usr/src

# tar -jxvf busybox-1.16.1.tar.bz2

# cd busybox-1.16.1

# make menuconfig

说明：

1、此处需要选择 Busybox Settings --> Build Options --> Build BusyBox as a static binary (no shared libs)，这样可以把Busybox编译成一个不使用共享库的静态二进制文件，从而避免了对宿主机的共享库产生依赖；但你也可以不选择此项，而完成编译后把其依赖的共享库复制至目标系统上的/lib目录中即可；这里采用后一种办法。

2、修改安装位置为/mnt/root;方法为：Busybox Settings --> Installation Options --> (./\_install) BusyBox installation prefix，修改其值为/mnt/sysroot。

# make install

安装后的文件均位于/mnt/sysroot目录中；但为了创建initrd，并实现让其启动以后将真正的文件系统切换至目标系统分区上的rootfs，您还需要复制一份刚安装在/mnt/sysroot下的busybox至另一个目录，以实现与真正的根文件系统分开制作。我们这里选择使用/mnt/temp目录；

# mkdir -pv /mnt/temp

# cp -r /mnt/sysroot/\* /mnt/temp

四、制作initrd

# cd /mnt/temp

1、建立rootfs：

# mkdir -pv proc sys etc/init.d tmp dev mnt/sysroot

2、创建两个必要的设备文件：

# mknod dev/console c 5 1

# mknod dev/null c 1 3

3、为initrd制作init程序，此程序的主要任务是实现rootfs的切换，因此，可以以脚本的方式来实现它：

# rm linuxrc

# vim init

添加如下内容：

#!/bin/sh

mount -t proc proc /proc

mount -t sysfs sysfs /sys

mdev -s

mount -t ext3 /dev/hda2 /mnt/sysroot

exec switch\_root /mnt/sysroot /init

给此脚本执行权限：

chmod +x init

4、制作initrd

# find . | cpio --quiet -H newc -o | gzip -9 -n > /mnt/boot/initrd.gz

五、建立真正的根文件系统

# cd /mnt/sysroot

1、建立rootfs：

# mkdir -pv proc sys etc/init.d tmp dev/pts boot var/log

2、创建两个必要的设备文件：

# mknod dev/console c 5 1

# mknod dev/null c 1 3

3、建立系统初始化脚本文件

# vim etc/init.d/rcS

添加如下内容：

#!/bin/sh

echo -e " Welcome to \033[31mToyLinux\033[0m "

echo -e "Remounting the root filesystem ..........[ \033[32mOK\033[0m ]"

mount -t proc proc /proc

mount -t sysfs sysfs /sys

mount -o remount,rw /

echo -e "Creating the files of device ............[ \033[32mOK\033[0m ]"

mdev -s

echo -e "Mounting the filesystem .................[ \033[32mOK\033[0m ]"

mount -a

swapon -a

echo -e "Starting the log daemon .................[ \033[32mOK\033[0m ]"

syslogd

klogd

echo -e "Configuring loopback interface ..........[ \033[32mOK\033[0m ]"

ifconfig lo 127.0.0.1/24

ifconfig eth0 172.16.100.9/16

# END

而后让此脚本具有执行权限：

chmod +x etc/init.d/rcS

4、配置init及其所需要inittab文件

# cd /mnt/sysroot

# mv linuxrc init

# vim etc/inittab

添加如下内容：

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

console::respawn:-/bin/sh

::ctrlaltdel:/sbin/reboot

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

5、为系统准备一个“文件系统表”配置文件/etc/fstab

# vim etc/fstab

添加如下内容：

sysfs /sys sysfs defaults 0 0

proc /proc proc defaults 0 0

/dev/hda1 /boot ext3 defaults 0 0

/dev/hda2 / ext3 defaults 1 1

6、由于在rcS文件中启动了日志进程，因此系统在运行中会产生大量日志并将其显示于控制台；这将会经常性的打断正在进行的工作，为了避免这种情况，我们这里为日志进程建立配置文件，为其指定将日志发送至/var/log/messages文件；

# vim etc/syslog.conf

添加如下一行：

\*.info /var/log/messages

六、好了，至此一个简易的基于内存运行的小系统已经构建出来了，我们接下来为此系统创建所需的引导程序

# grub-install --root-directory=/mnt /dev/hda

说明：此处的/dev/hda为目标系统所在的那块新磁盘；

接下来为grub建立配置文件：

# vim /mnt/boot/grub/grub.conf

添加类似如下内容：

default 0

timeout 3

color light-green/black light-magenta/black

title ToyLinux (2.6.34.1)

root (hd0,0)

kernel /bzImage ro root=/dev/hda2 quiet

initrd /initrd.gz

接下来将此块硬盘接入一个新的主机（这里使用的是虚拟机），启动一下并测试使用。

七、为新构建的ToyLinux启用虚拟控制台

这个可以通过宿主机来实现，也可以直接启动刚构建成功的小Linux进行配置。我们这里采用通过宿主机的方式（重新启动宿主机）：

# cd /mnt/sysroot

将 etc/inittab文件改为如下内容：

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

tty1::askfirst:/bin/sh

tty2::askfirst:/bin/sh

tty3::askfirst:/bin/sh

tty4::askfirst:/bin/sh

tty5::askfirst:/bin/sh

tty6::askfirst:/bin/sh

::ctrlaltdel:/sbin/reboot

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

好了，接下来就可以测试验正六个虚拟控制台的使用了。

八、尽管上述第七步已经实现了虚拟控制台，但其仍是直接进入系统，且系统没有用户帐号等安全设施，这将不利于系统的安全性。因此，接下来的这步实现为系统添加用户帐号（这里仍然基于宿主机实现）。

1、为目标主机建立passwd帐号文件

# cd /mnt/sysroot

# vim etc/passwd

添加如下内容：

root:x:0:0::/root:/bin/sh

而后为root用户创建“家”目录：

# mkdir root

2、为目标主机建立group帐号文件

# vim etc/group

添加如下内容：

root:x:0:

3、为目标主机建立shadow影子口令文件，这里采用直接复制宿主机的shadow文件中关于root口令行的行来实现

# grep "^root" /etc/shadow > etc/shadow

注：等目标主机启动时，root用户的口令也是宿主机的root用户的口令。您可以在目标主机启动以后再动手更改root用户的口令。

4、将 etc/inittab文件改为如下内容：

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

::respawn:/sbin/getty 9600 tty1

::respawn:/sbin/getty 9600 tty2

::respawn:/sbin/getty 9600 tty3

::respawn:/sbin/getty 9600 tty4

::respawn:/sbin/getty 9600 tty5

::respawn:/sbin/getty 9600 tty6

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

::ctrlaltdel:/sbin/reboot

好了，接下来就可以重新启动目标主机进行验正了。

九、在系统登录时提供banner信息

这个可以通过宿主机来实现，也可以直接在目标主机上进行配置。这里采用直接在目标主机上配置的方式：

# vi /etc/issue

添加如下内容：

Welcome to ToyLinux(http://marion.cublog.cn)...

Kernel \r

注：这里的内容可以根据你的需要进行修改。

十、在系统启动时为系统提供主机名称：

这个可以通过宿主机来实现，也可以直接在目标主机上进行配置。这里采用直接在目标主机上配置的方式：

1、创建保存主机名称的配置文件

# mkdir /etc/sysconfig

# vi /etc/sysconfig/network

添加如下内容：

HOSTNAME=marion.example.com

2、编辑系统初始化脚本，实现开机过程中设定主机名称

# vi /etc/init.d/rcS

在文件尾部添加如下行：

HOSTNAME=

[ -e /etc/sysconfig/network && -r /etc/sysconfig/network ] && source /etc/sysconfig/network

[ -z ${HOSTNAME} ] &&　HOSTNAME="localhost"

/bin/hostname ${HOSTNAME}