说明：本次构建均在虚拟机环境完成

准备工作

1. busybox(1.20.2)源码包、Linux(2.6.38.5)内核源码包
2. CentOS 5.11 作为宿主机
3. 在宿主机上添加额外一块硬盘。给宿主机添加一块硬盘并分区为hda1 和hda2，然后格式化为 ext3 文件系统，并分别挂载到宿主机/mnt/boot、/mnt/sysroot目录下

构建思路

1.busybox介绍

Busybox在单一的可执行文件中，模拟了很多Linux的命令。busybox适用于嵌入式系统。

2.目的系统构成

手动编译定制内核，作为目的系统的kernel，利用busybox制作initrd用来引导加载内核完成后挂载真正的roofs，以及用buxybox制作根文件系统。

3.基本流程

1. 手动编译 busybox 及 kernel（也可以使用宿主机的内核作为目的系统的内核）
2. 利用 busybox 制作 initrd 文件，提供根文件系统所在设备需要的驱动以及文件系统相关的驱动
3. 利用 busybox 制作 Linux 根文件系统
4. 安装引导加载内核的必要工具grub
5. 进行必要的配置
6. 为目的系统添加额外功能

编译 busybox 及准备 kernel

1.下载并解压缩 busybox 源码包

```
wget http://busybox.net/downloads/busybox-1.20.2.tar.bz2
tar -xf busybox-1.20.2.tar.bz
```

2.编译 busybox

说明：编译前需安装好必要的编译工具,编译安装目录在/root/works/busybox-1.20.2

```
#进入解压缩后的目录
cd busybox-1.20.2

#配置busybox编译文件
#此处选择 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
make menuconfig

#编译安装。此处，可能会遇到报错，缺少编译依赖的库文件或者busybox依赖于更新内核的头文件
#根据具体报错提示，判断并解决问题
make install
```

3.复制内核

此处利用宿主机的内核，也可以自己手动编译定制内核

注意：在手动编译内核时，将所需要的设备驱动模块和文件系统模块编译进内核。

```
#复制内核
cp /boot/vmlinuz-2.6.32-573.el6.x86_64 /mnt/boot/vmlinuz
```

制作 initrd 文件

说明：buxybox安装好的所有文件在`busybox-1.20.2`目录下`_install`目录内

1.创建一个临时目录用来制作initrd文件

```
mkdir -p /root/works/mkinitrd
```

2.复制_install目录下所有文件到/root/works/mkinitrd目录下

```
cp -ar /root/works/busybox-1.20.2/_install/* /rot/works/mkinitrd/
cd /root/works/mkinitrd
```

3.删除目录下不必要的linuxrc链接文件

```
rm linuxrc
```

4.创建必要的目录

```
mkdir -p dev etc lib/modules mnt/sysroot proc sys tmp
```

5.复制模块

initrd主要作用是内核加载完成后提供挂载真正的rootfs所需要的设备驱动和者文件系统模块，上面我们把rootfs所在分区格式化为ext3文件系统，内核利用initrd提供根文件系统所在设备的驱动，因为内核已经具有IDE的驱动功能，所以这里只需要复制识别文件系统的模块，要识别rootfs所在的分区，必须有ext3模块，所以复制ext3模块，

```
modprobe --show-depends ext3
cp /lib/modules/2.6.32-573.el6.x86_64/kernel/fs/jbd/jbd.ko lib/modules/
cp /lib/modules/2.6.32-573.el6.x86_64/kernel/fs/ext3/ext3.ko lib/modules/
```

6.提供init脚本

```
```

```
vim init
 #!/bin/sh
 echo "mounting proc and sys ..."
 mount -t proc proc /proc #挂载proc文件系统
 mount -t sysfs sysfs /sys #挂载sys文件系统

 echo "Load ext3 modules.."
 insmod /lib/modules/jbd.ko #加载模块
 insmod /lib/modules/ext3.ko

 echo "Detect and export hardware infomation..."
 mdev -s #探测额外硬件

 echo "mounting real rootfs to /mnt/sysroot..."
 mount -t ext3 /dev/sda2 /mnt/sysroot #挂载根文件系统到临时目录/mnt/sysroot

 echo "switch to real rootfs ..."
  exec switch_root /mnt/sysroot /sbin/init #切换根
```

7.修改init权限

```
chmod +x init
```

8.手动创建两个必要的设备文件

```
mknod dev/console c 5 1
mknod dev/null c 1 3
```

7.打包制作initrd

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

制作 Linux 根文件系统

```
#进入编译安装的目录
cd /root/works/busybox-1.20.2/_install

#复制根文件系统
cp -a * /mnt/sysroot/

#创建所需要的目录
cd /mnt/sysroot
mkdir proc sys dev tmp var/{log,lock,run} lib/modules
etc/rc.d/init.d root hmoe boot mnt/sysroot media -pv
```

安装 grub

```
#安装grub
grub-install --root-directory=/mnt /dev/sda

#添加grub配置文件
vim /mnt/boot/grub/grub.conf
 default=0
 timeout=5
  title Mini Linux (2.6.32)
  root(hd0,0)
  kernel /vmlinuz ro root=/dev/hda2
  initrd /initrd.gz
```

提供必要配置文件

1.提供etc/inittab

busybox的init程序不支持运行级别

```
cd /mnt/sysroot/
vim etc/inittab
 ::sysinit:/etc/rc.d/rc.sysinit
 console::respawn:-/bin/sh
 ::ctrlaltdel:/sbin/reboot #定义组合键
 ::shutdown:/bin/umount -a -r #执行shutdown命令，卸载所有文件系统
```

2.提供etc/fstab

```
vim etc/fstab
sysfs    /sys     sysfs    defaults     0 0
proc     /proc    proc     defaults     0 0
/dev/sda1    /boot    ext3     defaults     0 0
/dev/sda2    /    ext3     defaults     1 1
```

3.在根文件系统上创建两个必要的设备文件

```
mknod dev/console c 5 1
mknod dev/null c 1 3
```

4.创建etc/rc.d/rc.sysinit脚本

```
vim etc/rc.d/rc.sysinit
 #!/bin/sh
 echo -e "\tWelcome to  \033[34Mini Linux\033[0m"
 echo -e "Remounting the root filesystem ...[\033[32mOK\033[0m ]"
 mount -t proc proc /proc #busybox不能自动重新挂载这两个文件系统，所以在这而需要重新挂载
 mount -t sysfs sysfs /sys
 mount -t ext3 -o remount,rw /dev/hda2 /
 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

#修改权限
chmod +x etc/rc.d/rc.sysinit
```

添加额外功能

> 完成以上步骤，即可以开机测试，测试是否可以正常启动。

1.移植bash

执行此脚本移植bash程序及其所依赖的库。

```
#!/bin/bash
#
target=/mnt/sysroot

clearCmd() {
  if which $cmd &> /dev/null; then
        cmdPath=`which --skip-alias $cmd`
  else
        echo "No such command"
        return 5
  fi
}

cmdCopy() {
        cmdDir=`dirname $1`
        [ -d ${target}${cmdDir} ] || mkdir -p ${target}${cmdDir}
        [ -f ${target}${1} ] || cp $1 ${target}${cmdDir}
}

libCopy() {
        for lib in `ldd $1 | grep -o "/[^[:space:]]\{1,\}"`; do
                libDir=`dirname $lib`
```

```
                [ -d ${target}${libDir} ] || mkdir -p ${target}${libDir}
                [ -f ${target}${lib} ] || cp $lib ${target}${libDir}
        done
}

while true; do
  read -p "Enter a command: " cmd
  if [ "$cmd" == 'quit' ] ;then
        echo "quit"
        exit 0
  fi
  clearCmd $cmd
  [ $? -eq 5 ] && continue

  cmdCopy $cmdPath
  libCopy $cmdPath
done
```

```
#切换根目录，测试
chroot /mnt/sysroot
```

```
#修改系统默认启动shell
#修改/mntsysroot/etc/inittab
console::respawn:-/bin/bash
```

2.提供虚拟终端及用户登陆

修改etc/inittab文件

```
::sysinit:/etc/rc.d/rc.sysinit
::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
```

提供账号密码

```
grep -E "^root:" /etc/passwd > /mnt/sysroot/etc/passwd
grep -E "^root:" /etc/shadow > /mnt/sysroot/etc/shadow
grep -E "^root:" /etc/group > /mnt/sysroot/etc/group
```

3.提供主机名

提供一个配置文件etc/hostname

```
vim etc/hostname
HOSTNAME=xuekaixin
```

修改配置文件etc/rc.sysinit，并添加

```
vim etc/rc.sysinit
echo "set hostname ... "
[ -f /etc/hostname ] && ./etc/hostname
[ -z "$HOSTNAME" -o "HOSTNAME" == '(none)' ] && HOSTNAME=localhost
hostname $HOSTNAME
```

4.打印欢迎信息

```
cp /etc/issue /mnt/sysroot/etc/
```

5.提供网络功能

```
#查看网卡驱动模块
modinfo e1000

#复制模块
cp /lib/modules/2.6.18-398.el5/kernel/drivers/net/e1000/e1000.ko /mnt/sysroot/lib/modules/

#设置配置文件,添加
vim etc/rc.d/rc.sysinit

echo -e "Load ethernet card modules .."
insmod /lib/modules/e1000.ko

echo -e "set IP ... "
ifconfig lo 127.0.0.1/24
ifconfig eth0 192.168.1.109/24
```

文件系统修复

1.在所要修复的目录下打包所有文件

```
find . | cpio -H newc --quiet -o | gzip -9 > /path/to/file.gz
```

2.卸载设备

```
umount /dev/dev_name
#如果无法卸载，使用命令
fuser -km /dev/dev_name
```

3.重新格式化

```
mke2fs -j /dev/dev_name
```

4.重新挂载

```
mount /deb/dev_name /dir/to/dir_file
```

5.展开打包文件（进入目录）

```
zcat /path/to/file.gz | cpio -id
```

> **我们曾挥霍的一切在最高处必将回合！**