11.2 从零开始构建自己的U盘Linux

此方法比较复杂，但可以了解Linux开发的整个过程，其主要步骤如下：

(1)前期准备，包含软硬件的准备工作。

(2)Linux内核编译。

(3)构建Linux根文件系统，建立系统必需的目录、命令和设备。

(4)在U盘上安装系统引导程序grub。

11.2.1 前期准备

在做这个练习之前，需要准备以下内容：

1.准备开发主机

准备一台计算机，安装Linux操作系统(例如Redhat，Ubuntu等)，也可以在虚拟机上(例如VMWare)安装Linux操作系统。安装时，为了以后方便，可以选择安装全部组件。我们把这台装有Linux系统的计算机(或虚拟机)作为开发主机使用。

注意：(1)由于目前Ubuntu比较流行，以下操作示例均在Ubuntu下完成。Ubuntu下载地址http://www.ubuntu.org.cn/download。

(2)大多数操作需要root权限，由于Ubuntu默认为普通用户权限登录，可以在每个命令前添加sudo来实现使用root权限操作。

2.准备测试用计算机

测试U盘Linux的PC机必须支持USB硬盘启动方式，即把U盘作为硬盘来对待的启动方式(USB-HDD)。有些主板是把USB设备作为软盘方式(即USB-FDD)来启动系统的。目前，这些主板还无法完成该试验。

3.准备Linux的内核源代码

Linux的内核源代码可以从http://www.Linux.org上下载。另外，如果开发安装Linux系统时选择的是全部安装，在开发主机的/usr/src/目录中也会有Linux的源代码。

4.准备BusyBox工具

BusyBox工具中包含了七十多种Linux系统中常用的工具程序，利用BusyBox可以替代Linux系统中常用的一些工具和命令，例如ls，cp，rm，rmdir，mount，umount，init等。BusyBox中命令不仅丰富，而且占据很小的空间，同时它还提供面向嵌入式系统的应用。因此，在构建Linux系统时，特别是针对嵌入式Linux系统的应用中，使用BusyBox

取代常用的 Linux 命令非常有效。

BusyBox 的实质是提供了一个很小的可执行程序 BusyBox，通过对其的链接，可以建立其他常用的 Linux 系统命令。BusyBox 的具体使用方法如下：

(1)从 BusyBox 的官方网站 http://www.busybox.net/downloads 上下载 BusyBox 的源代码，例如 busybox-1.21.0.tar.bz2。将其放到/tmp 目录中。

(2)解压缩 busybox-1.21.0.tar.bz2

```
$ cd /tmp                                    //进入/tmp 目录
$ tar-xvjf busybox-1.21.0.tar.bz2            //解压缩
```

(3)进入 busybox-1.2.2 目录，修改 BusyBox 中的 init.c 源代码，具体操作如下：

```
$ cd /tmp/busybox - 1.21.0
$ vi init/init.c //编辑 init.c 文件
```

找到 init.c 文件中的以下代码：

```
#define INIT_SRCIPT "/etc/init.d/rcS
```

把其修改为：

```
#define INIT_SRCIPT "/etc/rc.d/rc.sysinit"
```

修改的目的是把系统执行的第一个程序设为:/etc/rc.d/rc.sysinit。如果不修改，也可以把以后建立的/etc/rc.d/rc.sysinit 文件改为/etc/init.d/rcS。

(4)对 BusyBox 进行配置，具体操作如下：

```
$ make defconfig      //使用默认配置，让 busybox 包含常用的工具和命令
$ make menuconfig     //进入人工配置菜单
```

进入手工配置菜单后，根据需要作一些修改。主要需要修改的选项说明如下：

```
BusyBox Settings→Build Options
    [ * ] Build BusyBox as a static binary (no shared libs)
```

这个选项能把 BusyBox 编译成静态链接的可执行文件，运行时可以不需要其他函数库，建议选上。

```
    [ ] Do you want to build busybox with a Cross Compiler
```

本选项设置是否把 BusyBox 用于嵌入式系统中，如果是，则需要设置交叉编译器。如果选中，会出现如下提示：

```
    (/usr/local/hybus-arm-Linux-R1.1/bin/arm-Linux-)Cross Compiler prefix
```

选中此行，可以输入自己定义的交叉编译器的前缀。例如交叉编译器为“/usr/arm/bin/arm-elf-gcc”，则在这里输入“/usr/arm/bin/arm-elf-”即可。

```
BusyBox Settings→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，这将覆盖系统原有的命令。选择这个选项后，make install 后会生成一个叫_install 的目录，里面有 BusyBox 和指向它的链接。

进入 Shell 选项，选择 ash 作为默认的 Shell 程序。如下：

```
Shells→Choose your default shell (ash)
    [ * ]ash
```

```
[ ]hush
[ ]lash
[ ]msh
```

在 BusyBox 的其他选项中可以选择包含那些 Linux 系统的命令，根据具体的需要进行选择。由于在手工配置(make menuconfig)之前，使用了 make defconfig 命令，因此，此时的 BusyBox 中已经包含了大部分的 Linux 系统常用的工具和命令。当然，如果用户希望包含 BusyBox 支持的全部命令，也可以使用 make allyesconfig 来进行配置。

(5)编译 BusyBox，命令如下：

```
$ make
```

(6)安装 BusyBox，命令如下：

```
$ make install
```

make install 完成后，会在/tmp/busybox-1.21.0/目录下生成 _install 目录，里面会建立 bin 和 sbin 子目录，其中包含 BusyBox 可执行文件和所有 BusyBox 支持命令对其的链接。通过察看/tmp/busybox-1.21.0/_install/bin/目录下的链接，用户可以清楚地看到 BusyBox 中究竟支持了哪些命令和工具。如果用户希望的命令没有出现在这个目录中，就需要重新配置、编译 BusyBox，让其支持。

11.2.2 编译 Linux 内核

从网上 http://www.kernel.org 下载一个 Linux 内核，放到开发主机上，解压缩之后就可以配置、编译内核了，具体操作可以按照以下步骤进行：

(1)进入 Linux 内核源码所在目录，使用“make menuconfig”命令配置 Linux 内核。

需要注意是由于要支持 U 盘启动，配置内核时必须选择以下内容：

①选择 Device Drivers→Block devices 下的 Loopback device support、RAMblock device support 等支持；

②选择 Device Drivers→SCSI Support 下的 SCSI device support、SCSI disk support、SCSI low-level drivers→Buslogic SCSI support 等支持；

③选择 Device Drivers→USB Support 下的 Support for Host-side USB、Preliminary USB device filesystem、USB Mass Storage support 支持；另外，还需要选中至少一个“Host Controller Driver(HCD)”，比如适用于 USBI.1 的“UHCI HCD support”或“OHCI HCD support”，适用于 USB2.0 的“EHCI HCD (USB2.0)Support”。如果拿不准的话把它们全部选中。

选择好需要编译的选项后，在主菜单中，选择最后一项 Save an Alternate Configuration File。

(2)使用“make dep”命令寻找依存关系，由系统决定需要编译哪些内容。

(3)使用“make clean”命令清除以前编译内核时生成的中间文件等。

(4)使用“make bzImage”命令生成压缩的 Linux 内核文件。生成的内核文件被命名为 bzImage，位于“../arch/i386/boot”目录下。

11.2.3　在 U 盘上建立根文件系统

1. 在 U 盘上面建立 Linux 分区和 ext2 文件格式

在对 U 盘进行分区之前必须要得到 U 盘在系统中的设备文件，U 盘在 Linux 系统中被识别为 SCSI 设备，因此系统分配给其的设备文件一般为 sda、sdb、sdc 等，如果系统中只有一个 SCSI 设备，则插入的第一个 U 盘的设备文件一般为/dev/sda。对于 VMWare 下安装的 Linux 系统而言，第一个 U 盘的设备文件一般为 /dev/sdb。实际的设备文件可以通过“fdisk-l”指令来查看。知道 U 盘的设备文件之后，就可以对 U 盘进行分区和格式化，其具体操作如下：

(1)把 U 盘插到开发主机上。

(2)使用“fdisk-l”命令查看 U 盘的设备文件。这里假设为/dev/sda1。

(3)使用 fdisk 在 U 盘上建立 Linux 分区，具体操作如下：

```
#fdisk / dev / sda      // 这里假设 U 盘的设备文件为 sda
Command(m for help):d      // 输入 d,删除旧的分区
Command(m for help):n      // 输入 n,建立新的分区
    e extended
    p     primary partition(1 - 4)
                  p              // 输入 p,选择建立主分区,回车
partition number(1 - 4):1   // 输入 1,建立 1 个分区
First cylinder (1 - 1019,default 1):   // 回车,选择默认
Last cylinder or + size or + sizeM or + sizeK (1 - 1019,default 1019): + 512M //由于现在的 U 盘都比较大,为了避免错误,这里可以输入 + 512M,在 U 盘上建立一个 512M 大小的分区
Command(m for help):p   // 输入 p,察看分区
Command(m for help):w   // 输入 w,保存并退出 fdisk
```

(4)格式化完成后，U 盘上会建立一个 Linux 分区。下面就可以在 U 盘上建立 ext2 文件系统，具体操作如下：

```
$ mkfs.ext2 /dev/sda1   //创建文件系统,这里假设 U 盘的设备文件为 sda1
```

2. 建立必需的目录

把 U 盘挂载到系统中，并且建立 /boot,/etc,/etc/rc. d,/proc,/tmp,/var,/dev,/mnt,/lib,/initrd 等系统必需的目录，具体操作如下：

```
$ mkdir /mnt/usb                  //建立 /mnt/usb 目录,用于挂载 U 盘
$ mount /dev/sda1 /mnt/usb      //挂载 U 盘到/mnt/usb 目录,假设 U 盘的设备文件为 sda1
$ cd /mnt/usb                     //进入 /mnt/sda 目录
$ mkdir boot etc etc/rc.d proc tmp var dev mnt lib initrd          //建立需要的目录
$ chmod 755 boot etc etc/rc.d proc tmp var dev mnt lib initrd      //改目录属性为可读写
```

这里没有建立 /bin 和/sbin 目录，这两个目录将直接从 BusyBox 的_install 复制过来。另外，如果使用 initrd 内核文件，也要创建 initrd 目录。

3. 建立必需的设备节点文件

进入 /mnt/usb/dev 目录，建立必需的设备节点文件，建立方法用两种，一是使用“cp-a”指令从系统的/dev 目录把需要的设备复制过来，另一种是使用 mknod 命令自己创建，自行创建的具体操作如下：

```
$ cd /mnt/usb/dev
```

(1)建立一般终端机设备

```
$ mknod tty c 5 0
$ mknod console c 5 1
$ chmod 666 tty console
```

(2)建立 VGA Display 虚拟终端机设备

```
$ mknod tty0 c 4 0
$ chmod 666 tty0
```

(3)建立 RAM disk 设备

```
$ mknod ram0 b 1 0
$ chmod 600 ram0
```

(4)建立 null 设备

```
$ mknod null c 1 3
$ chmod 666 null
```

4. 生成一些常见的命令和工具

文件系统中要包含一些常见的命令和工具，比如 ls、cp、rm、rmdir、init、ifconfig 等。用户可以复制原来系统中的这些命令，需要注意的是一定要把所用到的动态链接库(可以使用 ldd 命令查看)复制到/mnt/usb/lib 目录。

前面已经提过，这些常用 Linux 命令和工具会占用很多空间，有一种解决方法是使用 BusyBox 工具。BusyBox 工具中命令丰富，占用的空间又小，在本实验中将使用前面编译好的 BusyBox 工具。把 busybox-1.2.2/_install/目录下的 bin 目录和 sbin 目录复制到 U 盘的根目录下，命令如下：

```
$ cp-a-R-f /tmp/busybox-1.2.2/_install/* /mnt/usb/
```

使用-a 选项保证链接的正确性，使用-R 选项保证目录的正确复制，使用-f 选项进行强制覆盖。

在 BusyBox 工具中，还缺少 sh 命令，可以把 Linux 操作系统的 sh 命令复制过来，首先进入系统的/bin 目录，通过 ls-l 命令来查看 sh 命令，操作如下：

```
$ cd /bin
$ ls-l sh
```

发现 sh 命令实际上是 bash 命令的一个链接，再用 ldd 命令来查看 bash 的关联性：

```
$ ldd bash
```

发现 bash 需要/lib/libtermcap.so.2、/lib/libdl.so.2、/lib/tls/libc.so.6 和/lib/ld-Linux.so.2 库的支持，可以把这些库和 bash 复制到 U 盘中。具体操作如下：

```
$ cp /bin/bash /mnt/usb/bin
$ cp /lib/libtermcap.so.2 /mnt/usb/lib
$ cp /lib/libdl.so.2 /mnt/usb/lib
$ cp /lib/tls/libc.so.6 /mnt/usb/lib
$ cp /lib/ld-Linux.so.2 /mnt/usb/lib
$ cd /mnt/usb/bin
$ ln-s bash sh     //通过链接命令建立 sh 命令
```

至此，我们需要的命令已经建立完毕。

4. 建立一些必需的配置文件

Linux 系统在启动过程中还需要一些配置文件，比如/etc/rc.d/inittab、/etc/rc.d/rc.sysinit 和/etc/fstab 等。具体操作如下：

(1)建立 /mnt/usb/etc/rc.d/inittab 配置文件

```
$ vi /mnt/usb/etc/rc.d/inittab
```

添加以下内容：

```
::sysinit:/etc/rc.d/rc.sysinit
::askfirst:/bin/sh
```

如果是在窗口界面下操作，为了方便，可以使用 gedit 来创建上述文件，下同。

(2)建立 /mnt/usb/etc/rc.d/rc.sysinit 配置文件

```
$ vi /mnt/usb/etc/rc.d/rc.sysinit
```

添加以下内容：

```
$! /bin/sh
mount-a
```

(3)建立 /mnt/usb/etc/fstab 配置文件

```
$ vi /mnt/usb/etc/fstab
```

添加以下内容：

```
proc /proc proc defaults 0 0
```

(4)然后修改 inittab，rc.sysinit，fstab 这 3 个文件的权限

```
$ chmod 644 /mnt/usb/etc/inittab
$ chmod 755 /mnt/usb/etc/rc.d/rc.sysinit
$ chmod 644 /mnt/usb/etc/fstab
```

5. 复制 Linux 内核文件等到 U 盘中

复制编译好的内核文件到 U 盘：

```
$ cp × × × /arch/is36/boot/bzImage /mnt/usb/boot
```

如果使用了 initrd 内核文件，还需要复制其内核文件到 U 盘中，并把其后缀名改为 img，不改也可以。操作如下：

```
$ cp /tmp/initrd.gz /mnt/usb/boot/initrd. img
```

最后需要把 U 盘卸载下来，这样在/mnt/usb/中建立的目录和文件才会被保存到 U 盘中，可以使用“umount /mnt/usb”命令来卸载 U 盘。**这一步非常关键**，如果 U 盘没有

被 umount，则以上所有修改不会被保存到 U 盘上。

11.2.4 安装 grub 到 U 盘中

有了已经格式化好的 ext2 的文件系统，接下来就可以在这个文件系统上安装 Linux 的引导程序 grub 了。如果开发主机也是使用 grub 引导的，则在开发主机的/boot 目录下会安装有 grub 程序。如果没有，则需要自行下载之后进行安装，下载地址为 http://www.gnu.org/software/grub/。grub 的安装过程如下：

(1)首先，要将格式化好的优盘上的文件系统挂载到当前的 Linux 系统中。命令如下：

```
$ mount /dev/sda1 /mnt/usb
```

(2)建立 grub 所需要的目录，并将当前使用的 Linux 系统中的 grub 相关文件(/boot/grub/目录下的 stage1 和 stage2)复制到 U 盘的/usb/boot/grub 下。命令如下：

```
$ mkdir /mnt/usb/boot/grub
$ cp /boot/grub/stage* /mnt/usb/boot/grub/
```

(3)使用“grub”命令将 grub 引导程序安装在优盘上。具体命令如下：

```
$ grub
grub>root (hd1,0)
grub>setup (hd1)
grub>quit
```

上述操作中的 hd1 表明系统中已经有了一个硬盘，插入的 U 盘被标示为 hd1。类似这样的参数会随用户机器的硬盘数量和分布情况的不同而不同。

(4)在安装完 grub 后，还要对其进行配置。用户在 U 盘的/usr/boot/grub 目录下创建 grub.conf 文件(或者 grub.cfg，menu.lst，这和使用的 grub 版本和配置有关)，命令如下：

```
$ vi /mnt/usb/boot/grub/grub.conf
```

增加以下内容：

```
    default = 0
    timeout = 10
title MyUSBLinux-No-Use-initrd.img
    root (hd0,0)
    kernel /boot/bzImage ro root = /dev/sda1
title MyUSBLinux-Use-initrd.img
    root (hd0,0)
    kernel /boot/bzImageNoChange ro root = /dev/sda1
    initrd /boot/initrd.img
```

grub 中的 root (hd0,0)需要根据测试用计算机的具体情况更改，根据计算机上的 CMOS 设置的不同和硬盘数量的多少，这一项有可能是 root (hd1,0)或 root (hd2,0)等。

(5)最后使用“umount /mnt/usb”命令把 U 盘卸载掉即可。至此，一个可以在 U 盘

上独立运行的 Linux 操作系统就完成了。把 U 盘插到可以用 USB-HDD 方式启动的计算机上，设置好 CMOS 中的引导次序，就可以测试制作的 U 盘 Linux 系统了。

11.2.5　使用 initrd 内核作为根文件系统

上例中建立的根文件系统是直接存放在 U 盘上的，为了进一步节省空间，可以对准备好的根文件系统进行压缩存放。Linux 启动时，可以把压缩的根文件系统读到内存中，解压缩之后在进行加载。在嵌入式 Linux 系统设计中，通常会采用这种方式，例如前面使用的 initrd 内核实际上就是一个压缩的微型根文件系统，是否可以利用 initrd 内核来作为整个系统的根文件系统呢？答案是肯定的。initrd 内核实际上也是一个采用 Ramdisk 方式压缩的微型根文件系统，但是，在桌面 Linux 系统中，initrd 内核是作为一个临时文件系统使用的，其作用周期很短，实际的根文件系统一旦被加载，它就失去了作用。然而，在很多嵌入式系统的应用中，可以把 initrd 作为永久的根文件系统来使用。在本节，我们使用这种方式来购建一个使用 initrd 作为永久根文件系统的 Linux 操作系统。具体操作如下：

(1)创建一个 initrd 映像文件 initrdnew

创建 initrd 映像文件有多种方法，一是使用 Redhat 下建立 initrd 映像文件的专用命令 mkinitrd，具体操作如下：

```
$ mkinitrd /tmp/initrdnew.gz 2.4.20-8
$ gunzip /tmp/initrdnew.gz
$ mkinitrd /tmp/initrdnew.gz 2.4.20-8
```

另一种创建 initrd 的映像文件方法是采用通用的 loop 设备来制作，具体操作如下：

```
$ dd if = /dev/zero of = /tmp/ initrdnew bs = 1M count = 4
$ mkfs.ext2 /tmp/ initrdnew
```

上述命令创建了大小为 4M 的 initrd 映像文件 initrdnew，并且使用 mkfs.ext2 命令创建了 ext2 的文件系统。

(2)把 initrd 的映像文件 initrdnew 加载到系统中，具体操作如下：

```
$ mkdir /mnt/initrd                              //为加载上步得到的 initrd 文件作准备
$ mount-o loop /tmp/ initrdnew /mnt/initrd       //加载 initrd 文件
```

(3)建立必要的目录和命令

如果是使用 mkinitrd 创建的 initrd 映像文件，则其中将会有基本的目录结构。如果是使用 loop 设备创建的还需要建立一些基本的目录，例如/bin,/sbin,/boot,/etc,/proc,/tmp,/var,/dev,/mnt,/lib,/initrd 等系统必需的目录。具体操作如下：

```
$ cd /mnt/initrd
$ mkdir bin dev sys proc etc lib mnt
```

建立必要的命令：

```
$ cp /tmp/busybox-1.2.2/busybox /mnt/initrd/bin  //把 busybox 复制一份
$ cd /mnt/initrd/bin                             //进入 bin 目录
$ ln-s busybox ls                                //建立一个链接，使其具有 ls 命令
```

```
$ ln-s busybox cp
$ ln-s busybox ash
$ ln-s busybox mount
$ ln-s busybox echo
$ ln-s busybox ps
……
```

建立 sbin 目录：

```
$ cd /mnt/initrd
$ ln-s bin sbin
```

建立必要的设备：

```
$ cp-a /dev/console /mnt/initrd/dev
$ cp-a /dev/ramdisk /mnt/initrd/dev
$ cp-a /dev/ram0 /mnt/initrd/dev
$ cp-a /dev/null /mnt/initrd/dev
$ cp-a /dev/tty1 /mnt/initrd/dev
$ cp-a /dev/tty2 /mnt/initrd/dev
……
```

(4)建立并修改 Linuxrc 文件，操作如下：

```
$ vi /mnt/initrd/Linuxrc
```

加入的内容如下即可：

```
$ ! /bin/ash
/bin/ls
/bin/ash-login
```

(5)卸载 initrd，重新压缩生成 initrdnew. gz

```
$ umount /mnt/initrd
$ gzip-9 initrdnew
```

(6)U 盘的处理

参照以前的步骤，并在 U 盘上面建立 Linux 分区和 ext2 文件格式。建立/boot /boot/grub 目录。并且装上 grub。然后把以前编译好的 Linux 内核文件和上步得到的 initrdnew. gz 复制到 U 盘的/boot 目录中，创建和编辑 U 盘中的/boot/grub/grub. conf 文件(或者 grub. cfg)，操作如下：

```
$ vi /mnt/usb/boot/grub/grub.conf
```

输入以下内容：

```
default = 0
timeout = 10
title InitrdLinux
root (hd0,0)
kernel /boot/bzImage ro root = /dev/ram0 init = /Linuxrc rw
initrd /boot/initrdnew.gz
```

保存退出，把 U 盘卸载下来，整个制作过程结束。一个可以在 U 盘上独立运行的采

用 Ramdisk 方式的 Linux 操作系统就完成了，把 U 盘插到可以用 USB-HDD 方式启动的计算机上，设置好 CMOS 中的引导次序，就可以测试这个 Linux 系统了。

习　题

根据书上的例子，采用不同的方法，完成一个 U 盘 Linux 系统的制作与调试。