mx53 根文件系统的构建

这篇文章祥解了 mx53 根文件系统的构建，考虑到嵌入式的应用，这里使用 busybox 作为基础搭建根文件系统。版本为 1.20.2，可从官网直接下载。具体的搭建步骤如下：

1、 解压源码包

tar jxvf　busybox-1.20.2.tar.bz2

2、 配置 busybox

修改 Makefile，指定目标平台和交叉编译工具链。

将 Makefile 中的 Line 191 左右的 ARCH ?= $(SUBARCH)注释掉，在其后新增一行，如下：

ARCH = arm

```
187
188
189
190
191 #ARCH ?= $(SUBARCH)
192 ARCH = arm
193
194 # Architecture as present in compile.h
195 UTS_MACHINE := $(ARCH)
196
197 # SHELL used by kbuild
198 CONFIG_SHELL := $(shell if [ -x "$$BASH" ]
199           else if [ -x /bin/bash ]; then e
```

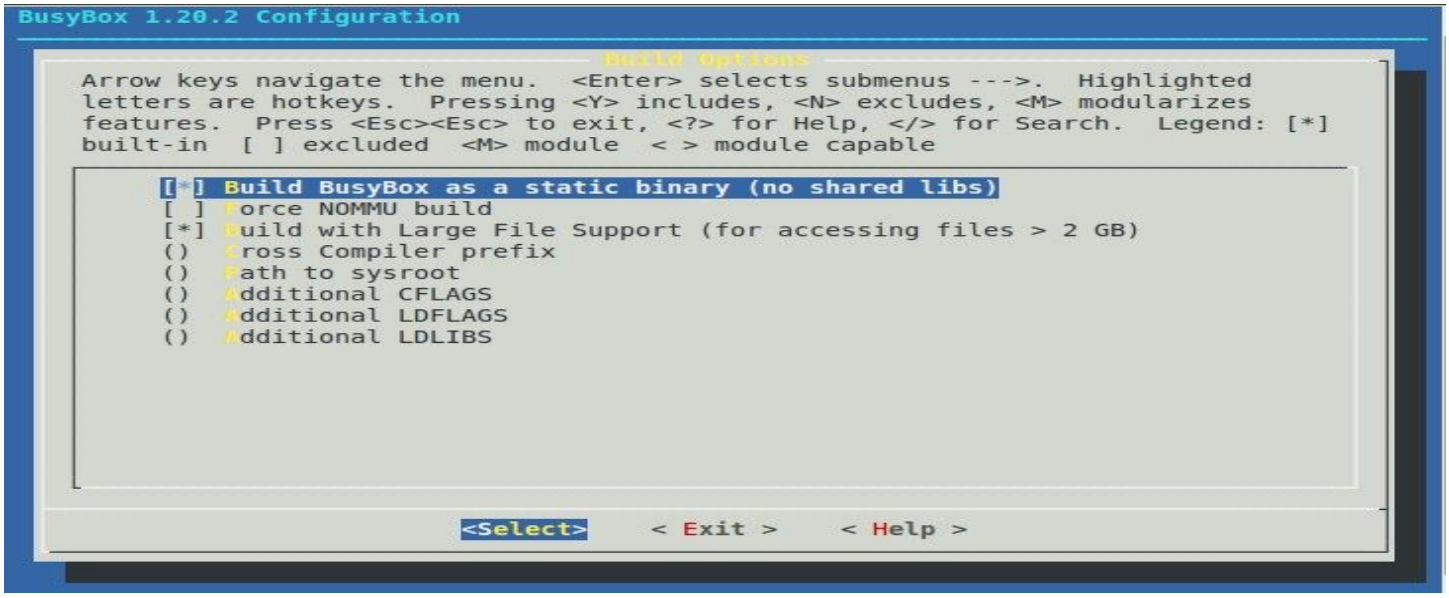
将 Makefile 中的 Line 164 左右的 CROSS_COMPILE ?= 注释掉，在其后新增一行，如下：

CROSS_COMPILE = /usr/local/arm/4.3.2/bin/arm-linux-

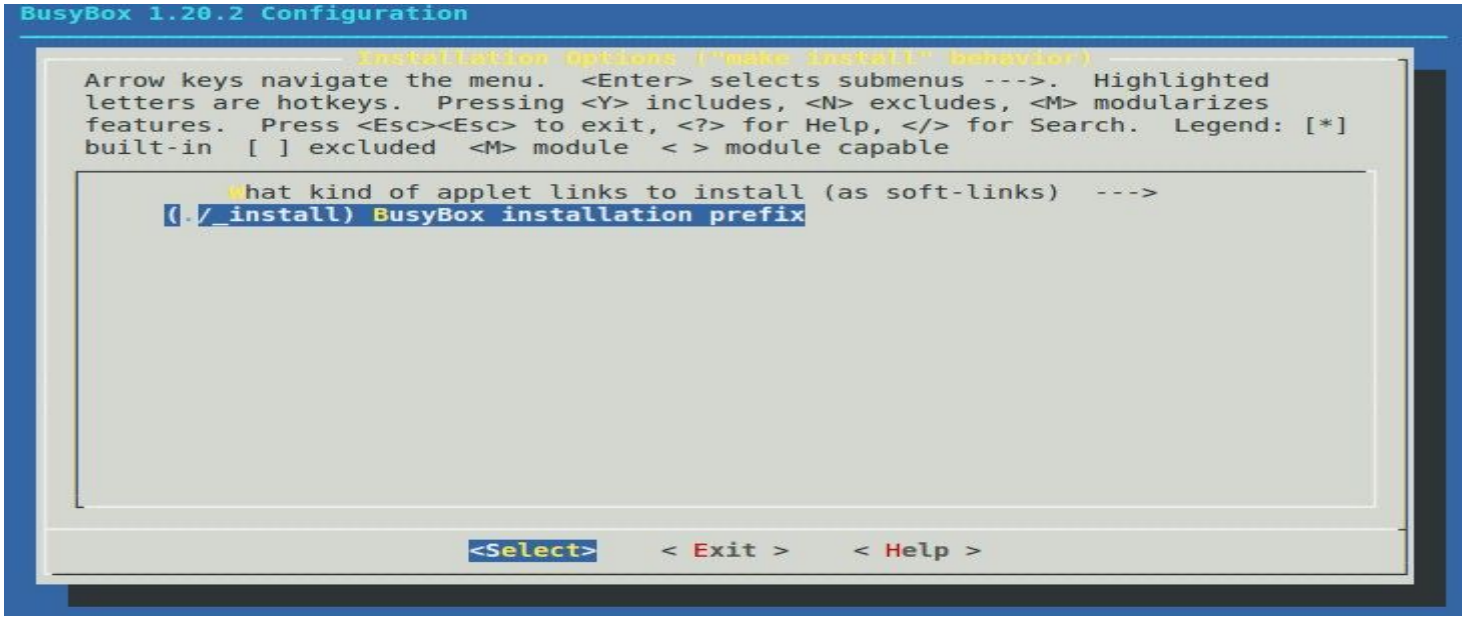
```
159 # make CROSS_COMPILE=ia64-linux-
160 # Alternatively CROSS_COMPILE can be set in the environment.
161 # Default value for CROSS_COMPILE is not to prefix executable
162 # Note: Some architectures assign CROSS_COMPILE in their arch
163
164 #CROSS_COMPILE ?=
165 CROSS_COMPILE = /usr/local/arm/4.3.2/bin/arm-linux-
166 # bbox: we may have CONFIG_CROSS_COMPILER_PREFIX in .config,
167 # and it has not been included yet... thus using an awkward s
168 ifeq ($(CROSS_COMPILE),)
169 CROSS_COMPILE := $(shell grep ^CONFIG_CROSS_COMPILER_PREFIX .
170 CROSS_COMPILE := $(subst CONFIG_CROSS_COMPILER_PREFIX=,,$(CRO
```

执行 make menuconfig，进行图形化配置。

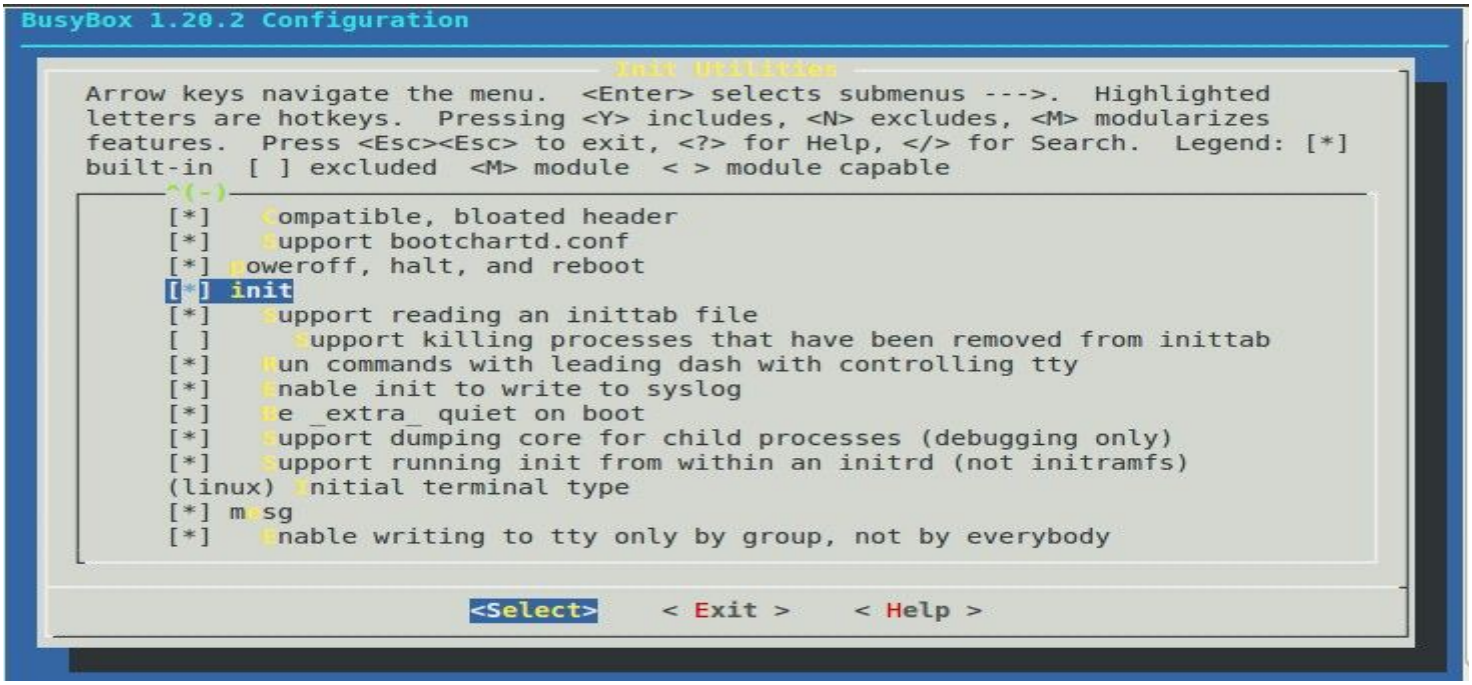
将 busybox 配置成静态编译，虽然生成的文件有点大，但是解决了动态库依赖的问题。路径为 Busybox Settings->Build Options，[*]Build BusyBox as a static binary(no shared libs)。



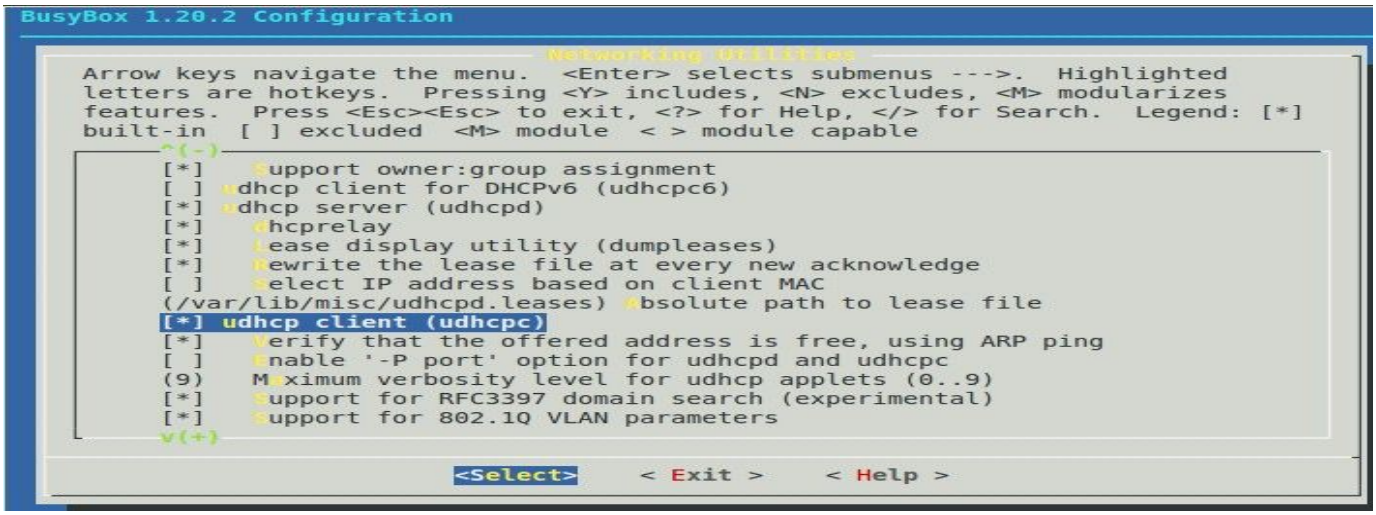
配置 make install 的动作路径，一定要配置，否则会安装到开发机的/usr 路径下的！路径为 Busybox Settings->Installation Options(“make install” behavior)，将 Busybox installation prefix 输入为 ./_install，这样生成的映像文件及链接文件会在根目录下的_install 文件夹下的。同时选择小应用程序的类型为“soft-links”。



配置 busybox 支持 init 进程，即内核启动后的第 1 个进程。这样 init 进程会去读取/etc/inittab 文件的。



配置支持 dhcp client，路径为 Networking Utilities。



根据需要选择一些关键的子功能，如 ifconfig、insmod/rmmod、pwd、ls、find 等等这些标准的 Linux 命令。

执行编译 make

```
zhangshaoyan@ubuntu:~/i.mx53/busybox-1.20.2$ make
  SPLIT   include/autoconf.h -> include/config/*
  GEN     include/bbconfigopts.h
  HOSTCC  applets/usage
applets/usage.c: In function ‘main’:
applets/usage.c:52: warning: ignoring return value of ‘write’, declared with attrib
_unused_result
  GEN     include/usage_compressed.h
  HOSTCC  applets/applet_tables
applets/applet_tables.c: In function ‘main’:
applets/applet_tables.c:144: warning: ignoring return value of ‘fgets’, declared wi
bute warn_unused_result
  GEN     include/applet_tables.h
  CC      applets/applets.o
  LD      applets/built-in.o
  HOSTCC  applets/usage_pod
applets/usage_pod.c: In function ‘main’:
applets/usage_pod.c:74: warning: format not a string literal and no format argument
  CC      libbb/appletlib.o
  AR      libbb/lib.a
```

执行安装 make install （说白了就是将编译生成的可执行文件拷贝到./_install 目录下。

```
./_install//usr/sbin/nbd-client -> ../../bin/busybox
./_install//usr/sbin/ntpd -> ../../bin/busybox
./_install//usr/sbin/popmaildir -> ../../bin/busybox
./_install//usr/sbin/rdate -> ../../bin/busybox
./_install//usr/sbin/rdev -> ../../bin/busybox
./_install//usr/sbin/readprofile -> ../../bin/busybox
./_install//usr/sbin/sendmail -> ../../bin/busybox
./_install//usr/sbin/setfont -> ../../bin/busybox
./_install//usr/sbin/setlogcons -> ../../bin/busybox
./_install//usr/sbin/svlogd -> ../../bin/busybox
./_install//usr/sbin/telnetd -> ../../bin/busybox
./_install//usr/sbin/ubiattach -> ../../bin/busybox
./_install//usr/sbin/ubidetach -> ../../bin/busybox
./_install//usr/sbin/ubimkvol -> ../../bin/busybox
./_install//usr/sbin/ubirmvol -> ../../bin/busybox
./_install//usr/sbin/ubirsvol -> ../../bin/busybox
./_install//usr/sbin/ubiupdatevol -> ../../bin/busybox
./_install//usr/sbin/udhcpd -> ../../bin/busybox


----------------------------------------------------
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.
----------------------------------------------------

zhangshaoyan@ubuntu:~/i.mx53/busybox-1.20.2$
```

现在，我们去./_install 目录下看看

```
zhangshaoyan@ubuntu:~/i.mx53/busybox-1.20.2/_install$ ls -l
total 12
drwxr-xr-x 2 zhangshaoyan zhangshaoyan 4096 2012-12-18 23:02 bin
lrwxrwxrwx 1 zhangshaoyan zhangshaoyan   11 2012-12-18 23:02 linuxrc -> bin/busybox
drwxr-xr-x 2 zhangshaoyan zhangshaoyan 4096 2012-12-18 23:02 sbin
drwxr-xr-x 4 zhangshaoyan zhangshaoyan 4096 2012-12-03 00:48 usr
zhangshaoyan@ubuntu:~/i.mx53/busybox-1.20.2/_install$
```

好了，现在就算编译成功了。

3、 建立基本的根文件系统目录结构

mkdir bin sbin etc etc/init.d lib dev proc sys tmp usr usr/bin usr/share usr/sbin usr/local -p

```
zhangshaoyan@ubuntu:~/i.mx53/nfsrootfs$ ls
bin  dev  etc  lib  linuxrc  proc  sbin  sys  tmp  usr
zhangshaoyan@ubuntu:~/i.mx53/nfsrootfs$
```

将编译 busybox 时生成的_install 目录下所有的东东都拷贝过来。

4、 建立 init 进程配置文件

init 进程是内核启动后的第 1 个进程，它会去读取/etc/inittab 文件决定如何动作。这里我们建立一个最基本的 inittab 文件，如下：

```
#/etc/inittab
::sysinit:/etc/init.d/rcS
::respawn:-/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r

~
~
```

这其中最关键的就是::respawn:-/bin/sh，表示当终端退出后，init 会重新执行/bin/sh，即进入 shell。

另一个比较关键的就是::sysinit:/etc/init.d/rcS，即 init 进程启动后，会调用 rcS 进行初始化，所以我们要写好这个 rcS。

5、系统初始化脚本 rcS

```
#!/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
export PATH
#call /etc/fstab
mount -a
~
~
~
```

这个脚本的内容很简单，一个是设置 PATH 环境变量，另一个是调用 mount -a，这句会让 mount 程序去读取/etc/fstab 文件，决定什么文件系统会被挂载。

5、文件系统挂载文件/etc/fstab

```
proc    /proc   proc    defaults        0       0
sysfs   /sys    sysfs   defaults        0       0
~
```

这个文件就是声明了挂载 proc 和 sysfs 文件系统，用于将内核数据导出。方便查看。

6、拷贝必要的 C 库文件

为了简单起见，我们就不编译 C 库了，而直接使用交叉编译工具链提供的动态库文件。将 arm-linux-4.3.2 提供的 libc 拷贝到根文件系统的 lib 目录下。

交叉编译工具链库文件所在路径为：/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/lib。

```
zhangshaoyan@ubuntu:/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/lib$ ls
ld-2.8.so               libdl-2.8.so          libnss_dns.so.2       libresolv-2.8.so
ld-linux.so.3           libdl.so.2            libnss_files-2.8.so   libresolv.so.2
libanl-2.8.so           libgcc_s.so           libnss_files.so.2     librt-2.8.so
libanl.so.1             libgcc_s.so.1         libnss_hesiod-2.8.so  librt.so.1
libBrokenLocale-2.8.so  libm-2.8.so           libnss_hesiod.so.2    libSegFault.so
libBrokenLocale.so.1    libmemusage.so        libnss_nis-2.8.so     libthread_db-1.0.so
libc-2.8.so             libm.so.6             libnss_nisplus-2.8.so libthread_db.so.1
libcidn-2.8.so          libnsl-2.8.so         libnss_nisplus.so.2   libutil-2.8.so
libcidn.so.1            libnsl.so.1           libnss_nis.so.2       libutil.so.1
libcrypt-2.8.so         libnss_compat-2.8.so  libpcprofile.so
libcrypt.so.1           libnss_compat.so.2    libpthread-2.8.so
libc.so.6               libnss_dns-2.8.so     libpthread.so.0
zhangshaoyan@ubuntu:/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/lib$
```

7、打包根文件系统

在根文件系统目录下执行 tar cvfj ../rootfs.tar.bz2 *

```
zhangshaoyan@ubuntu:~/i.mx53/nfsrootfs$ ls
bin  dev  etc  lib  linuxrc  proc  sbin  sys  tmp  usr
zhangshaoyan@ubuntu:~/i.mx53/nfsrootfs$ tar cvfj ../rootfs.tar.bz2 *
```

这样在上层目录就会生成 rootfs.tar.bz2 文件，使用 MFG Tools 烧写即可。

8、完毕！

shell.albert

2012/12/14