Creating the Root file system

- ✓ In order to build a root filesystem, you need a device that is large enough to hold all the files
- ✓ If you have an unused hard disk space than use a *loopback device*, which allows a disk file to be treated as a block device
- ✓ Creating a loopback device
  1. dd if=/dev/zero of=/tmp/myfs bs=<block_size> count=<no_blocks>
  2. losetup /dev/loop0 /tmp/myfs
- ✓ Format the loopback device
  1. mkfs.ext2 /dev/loop0
- ✓ Mount the loopback device
  1. mount -t ext2 /dev/loop0 /media

Creating the Root file system

- ✓ Populate your root filesystem with minimum set of directories
- ✓ Populating /bin, /sbin, /usr/bin, /usr/sbin directories with busybox
- ✓ Why busybox
  - ✓ Most Unix command line utilities within a single executable!
  - ✓ Sizes less than < 500 KB (statically compiled with uClibc) or less than
  - ✓ 1 MB (statically compiled with glibc).
  - ✓ Easy to configure which features to include.
  - ✓ The best choice for Small and medium size embedded systems

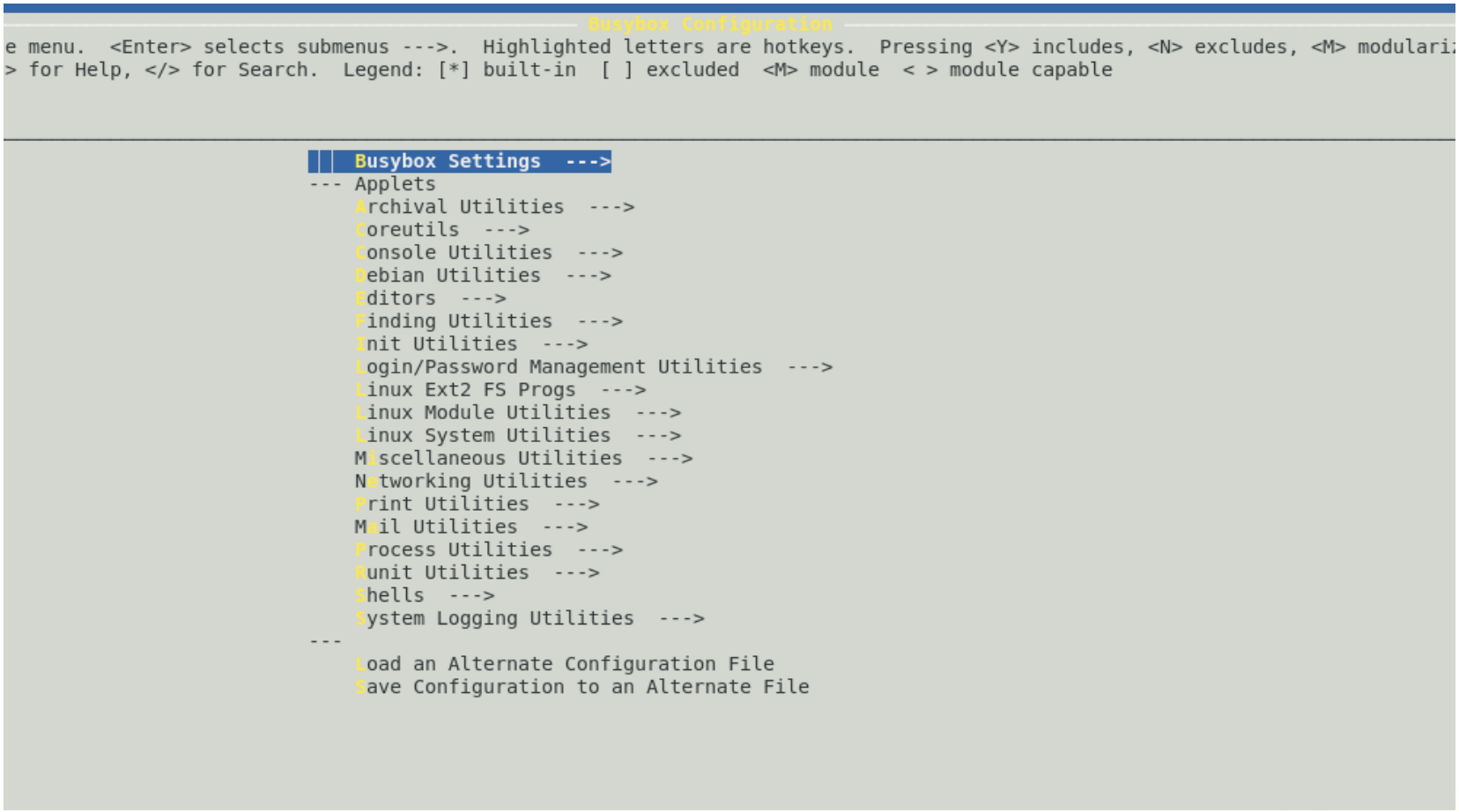
Download busybox from http://www.busybox.net/

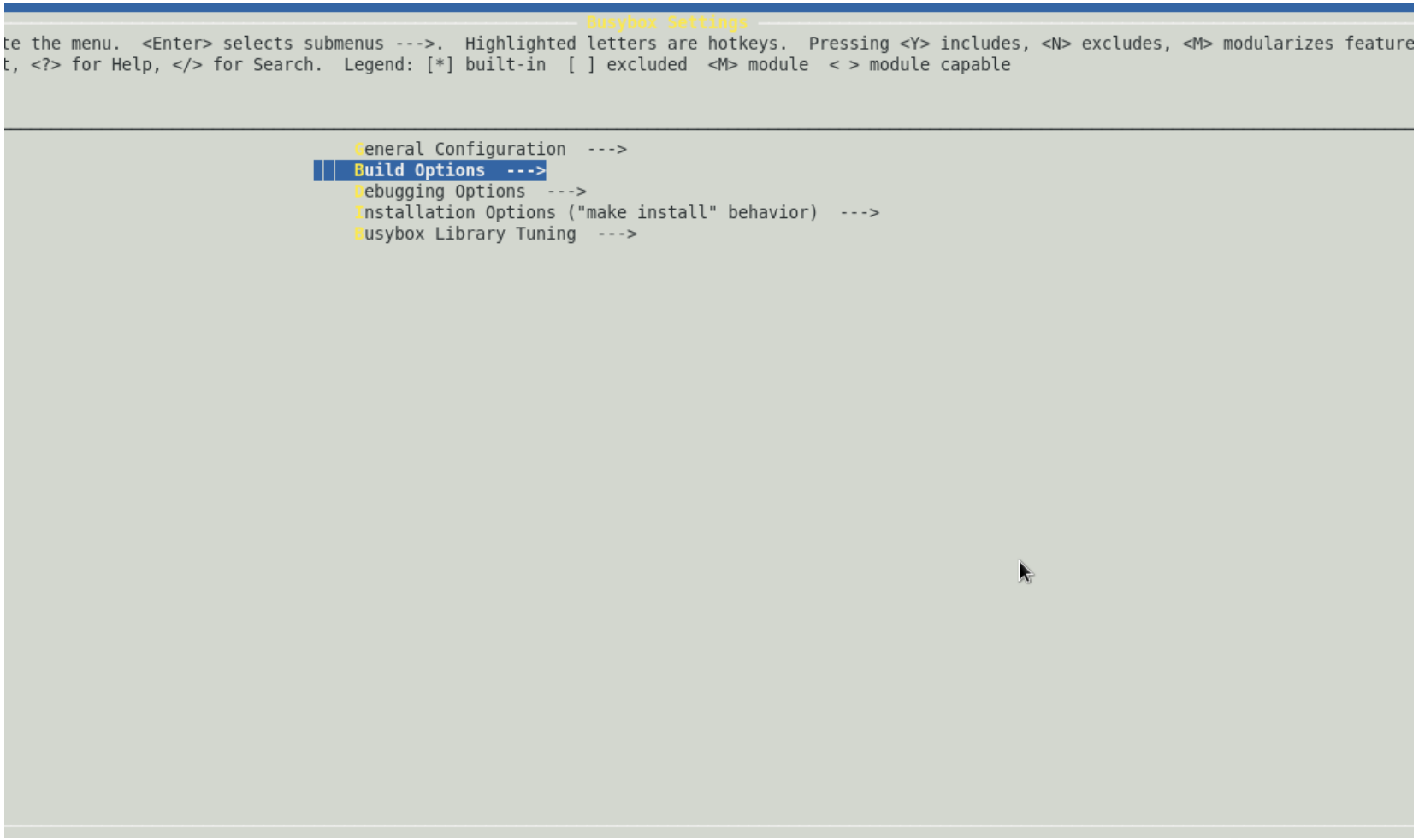
tar xvf BusyBox 1.18.1.tar.bz2

make menuconfig

export CROSS_COMPILE=$(CROSSTOOL_PATH)/arm-linux-

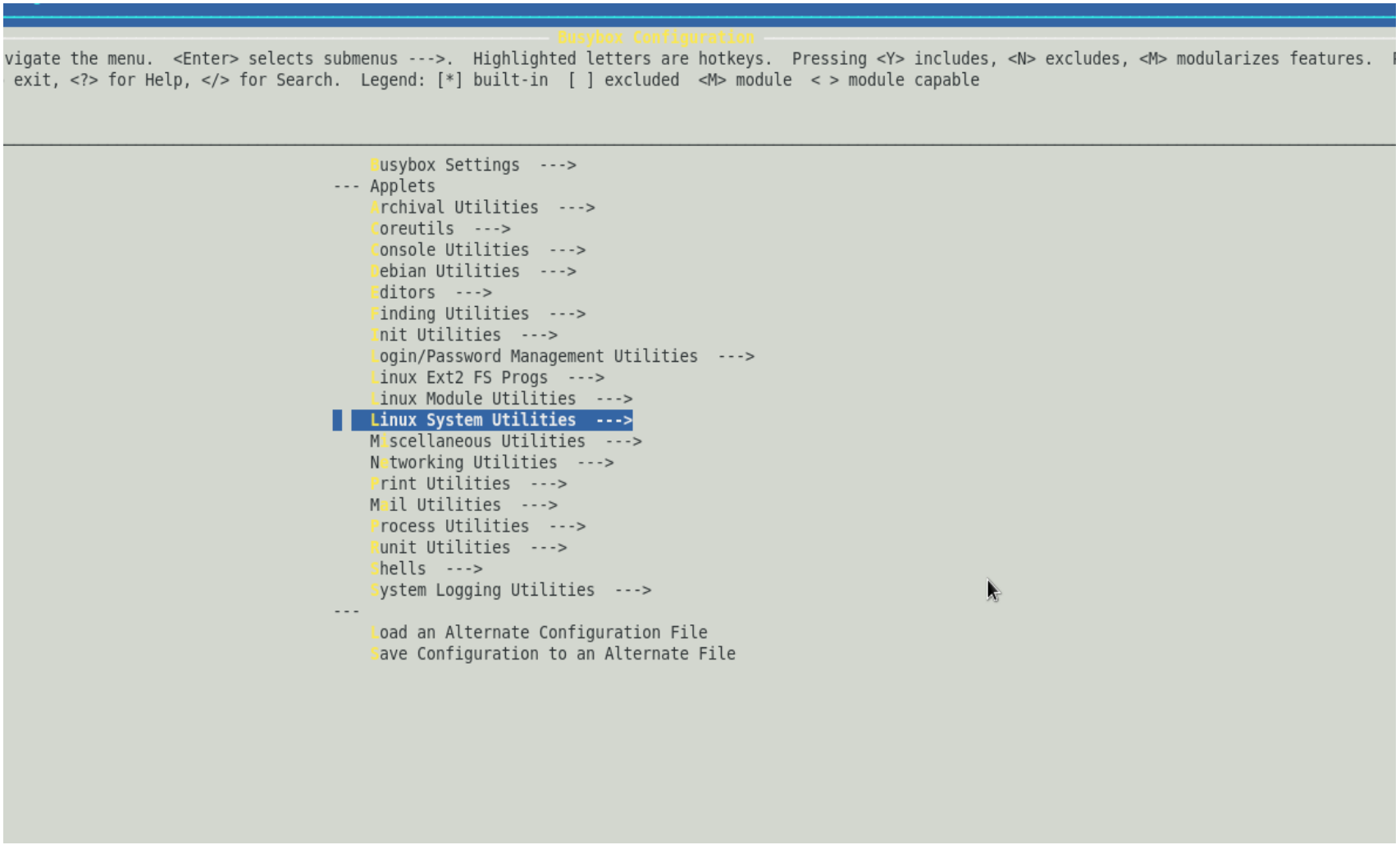
make



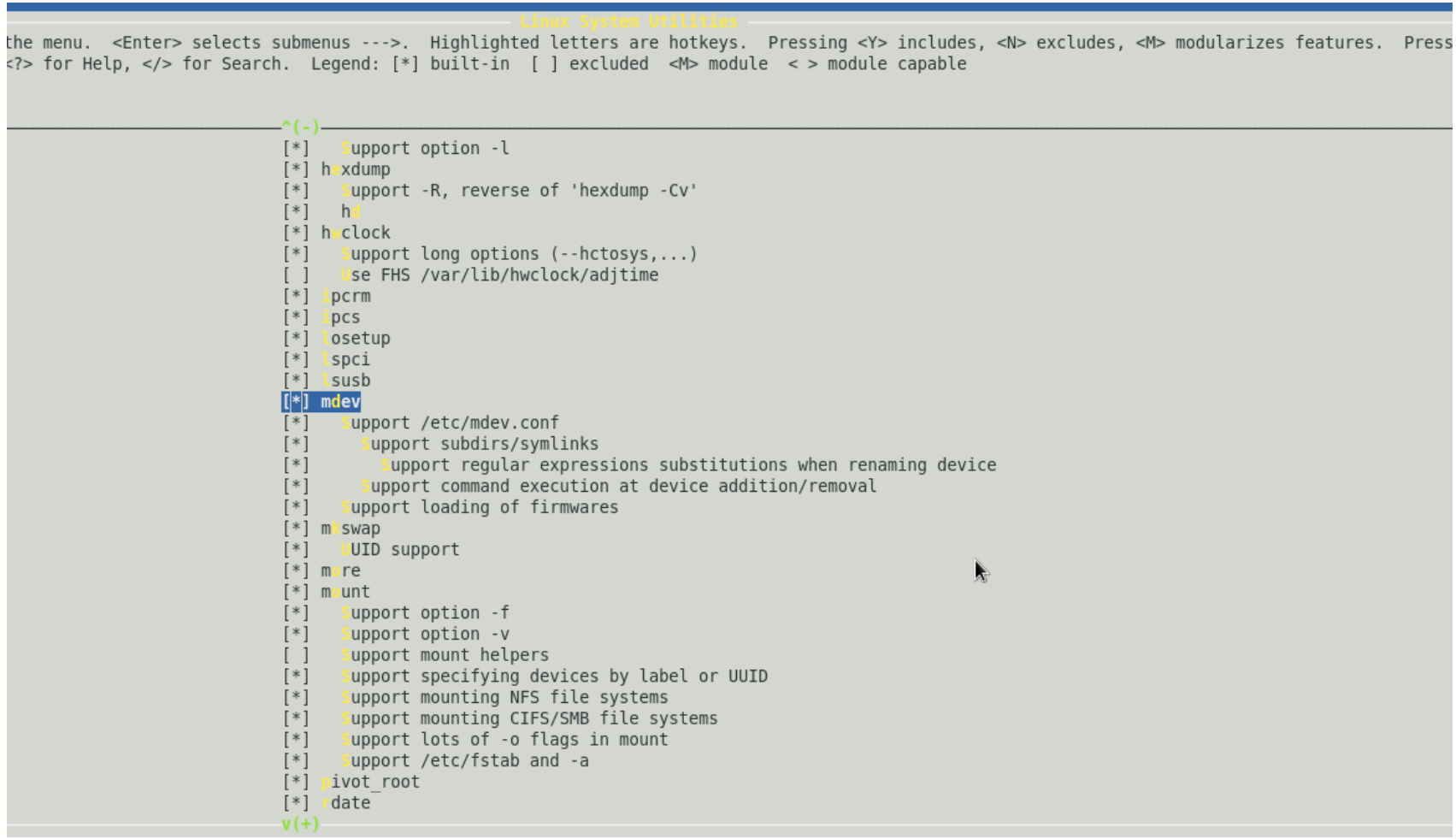

Busybox Settings
te the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes feature
t, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
General Configuration --->
Build Options --->
Debugging Options --->
Installation Options ("make install" behavior) --->
Busybox Library Tuning --->

```
                                  Build Options
the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.
<?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable


                [*] Build BusyBox as a static binary (no shared libs)
                [ ] Force NOMMU build
                [ ] Build with Large File Support (for accessing files > 2 GB)
                ()  Cross Compiler prefix
                ()  Additional CFLAGS
```


Busybox Configuration
vigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features.
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Busybox Settings --->
--- Applets
Archival Utilities --->
Coreutils --->
Console Utilities --->
Debian Utilities --->
Editors --->
Finding Utilities --->
Init Utilities --->
Login/Password Management Utilities --->
Linux Ext2 FS Progs --->
Linux Module Utilities --->
Linux System Utilities --->
Miscellaneous Utilities --->
Networking Utilities --->
Print Utilities --->
Mail Utilities --->
Process Utilities --->
Runit Utilities --->
Shells --->
System Logging Utilities --->

Load an Alternate Configuration File
Save Configuration to an Alternate File


Linux System Utilities
the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press
<?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
^(-)
[*] Support option -l
[*] hexdump
[*] Support -R, reverse of 'hexdump -Cv'
[*] hd
[*] hwclock
[*] Support long options (--hctosys,...)
[ ] Use FHS /var/lib/hwclock/adjtime
[*] ipcrm
[*] ipcs
[*] losetup
[*] lspci
[*] lsusb
[*] mdev
[*] Support /etc/mdev.conf
[*] Support subdirs/symlinks
[*] Support regular expressions substitutions when renaming device
[*] Support command execution at device addition/removal
[*] Support loading of firmwares
[*] mkswap
[*] UUID support
[*] more
[*] mount
[*] Support option -f
[*] Support option -v
[ ] Support mount helpers
[*] Support specifying devices by label or UUID
[*] Support mounting NFS file systems
[*] Support mounting CIFS/SMB file systems
[*] Support lots of -o flags in mount
[*] Support /etc/fstab and -a
[*] pivot_root
[*] rdate
v(+)

make install

Copy the output folders of bustbox to the root filesystem using

rsync –a _install/* /media

Create the following directories

mkdir dev, etc, lib, proc, root, sys, tmp

chmod 766 dev, etc, lib, proc, root, sys, tmp

Create the ***inittab*** file under ***/etc*** directory

#vi inittab and type the following

```
# Startup the system
null::sysinit:/etc/init.d/rcS

ttyS0::respawn:-/bin/sh

# Stuff to do before rebooting
null::shutdown:/bin/umount -a -r
```

Create the following directories under ***/etc***

```
# mkdir  etc/init.d
```

Create ***rcS*** file under ***etc/init.d*** directory

vi rcS and type the following

```
#!/bin/sh
echo -n " Mounting /proc          : "
mount -n -t proc /proc /proc

echo -n " Mounting /sys           : "
mount -n -t sysfs sysfs /sys

echo -n " Mounting /dev           : "
mount -n -t tmpfs mdev /dev
```

vi rcS

```
#  ----------------------------------------
#  Enabling hot-plug
#  ----------------------------------------
echo "/sbin/mdev" > /proc/sys/kernel/hotplug

mdev -s

#  ----------------------------------------
#  Set PATH
#  ----------------------------------------
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin

#  ----------------------------------------
#  Set ip address
#  ----------------------------------------
/sbin/ifconfig lo 127.0.0.1 up
/sbin/ifconfig eth0 192.168.1.1 up
```

Create the following device files under **/dev**

```
# mknod  console  c 5  1
# mknod  null  c 1  3
# mknod  ttyS0  c 4  64
```

Converting ext2 files system to jffs2 file system

```
# mkfs.jffs2  --pad=0x4000
              --eraseblock=0x4000
              -l  --root=/media –o my_file.bin
```

Thank you