08-1 BusyBox

The Swiss Army Knife of Embedded Linux

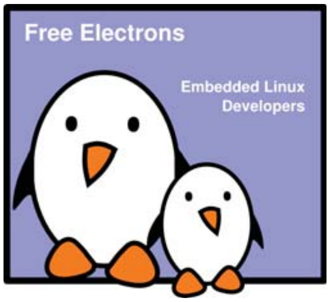
Chapters 9 and 10

- 9: File Systems
  - Partitions
  - ext2, ext3, ext4
  - ReiserFS
  - JFF2
  - NFS
  - /proc, /sys
- 10: Memory Technology Devices (MTD) Subsystem
  - Supports memory-like devices such as Flash
  - Device driver layer that provides an API for raw flash devices
  - Neither *block* nor *char*

Embedded Linux system development

BusyBox

Thomas Petazzoni
Michael Opdenacker
Free Electrons



Latest update: 10/27/2014,
Document sources, updates and translations:
http://free-electrons.com/docs/busybox
Corrections, suggestions, contributions and translations are welcome!

Why Busybox?

- A Linux system needs a basic set of programs to work
  - An init program
  - A shell
  - Various basic utilities for file manipulation and system configuration
- In normal Linux systems, those programs are provided by different projects
  - `coreutils`, `bash`, `grep`, `sed`, `tar`, `wget`, `modutils`, etc. are all different projects
  - A lot of different components to integrate
  - Components not designed with embedded systems constraints in mind: they are not very configurable and have a wide range of features
- Busybox is an alternative solution, extremely common on embedded systems

General purpose toolbox: BusyBox

- Rewrite of many useful Unix command line utilities
  - Integrated into a single project, which makes it easy to work with
  - Designed with embedded systems in mind: highly configurable, no unnecessary features
- All the utilities are compiled into a single executable, `/bin/busybox`
  - Symbolic links to `/bin/busybox` are created for each application integrated into Busybox
- For a fairly featureful configuration, less than 500 KB (statically compiled with uClibc) or less than 1 MB (statically compiled with glibc).
- http://www.busybox.net/

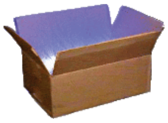
BusyBox commands!

```
Currently defined functions:
[, [[, acpid, add-shell, addgroup, adduser, adjtimex, arp, arping, ash, awk, base64, basename, beep, blkid,
blockdev, bootchartd, brctl, bunzip2, bzcat, bzip2, cal, cat, catv, chat, chattr, chgrp, chmod, chown,
chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, conspy, cp, cpio, crond, crontab, cryptpw,
cttyhack, cut, date, dc, dd, deallocvt, delgroup, deluser, depmod, devmem, df, dhcprelay, diff, dirname,
dmesg, dnsd, dnsdomainname, dos2unix, du, dumpkmap, dumpleases, echo, ed, egrep, eject, env, envdir,
envuidgid, ether-wake, expand, expr, fakeidentd, false, fatattr, fbset, fbsplash, fdflush, fdformat, fdisk,
fgconsole, fgrep, find, findfs, flock, fold, free, freeramdisk, fsck, fsck.minix, fstrim, fsync, ftpd,
ftpget, ftpput, fuser, getopt, getty, grep, groups, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid,
hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init, insmod, install,
ionice, iostat, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd_mode, kill, killall,
killall5, klogd, last, less, linux32, linux64, linuxrc, ln, loadfont, loadkmap, logger, login, logname,
logread, losetup, lpd, lpq, lpr, ls, lsattr, lsmod, lsof, lspci, lsusb, lzcat, lzma, lzop, lzopcat, makedevs,
makemime, man, md5sum, mdev, mesg, microcom, mkdir, mkdosfs, mke2fs, mkfifo, mkfs.ext2, mkfs.minix,
mkfs.vfat, mknod, mkpasswd, mkswap, mktemp, modinfo, modprobe, more, mount, mountpoint, mpstat, mt, mv,
nameif, nanddump, nandwrite, nbd-client, nc, netstat, nice, nmeter, nohup, nslookup, ntpd, od, openvt,
passwd, patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill, pmap, popmaildir, poweroff,
powertop, printenv, printf, ps, pscan, pstree, pwd, pwdx, raidautorun, rdate, rdev, readahead, readlink,
readprofile, realpath, reboot, reformime, remove-shell, renice, reset, resize, rev, rm, rmdir, rmmod, route,
rpm, rpm2cpio, rtcwake, run-parts, runlevel, runsv, runsvdir, rx, script, scriptreplay, sed, sendmail, seq,
setarch, setconsole, setfont, setkeycodes, setlogcons, setserial, setsid, setuidgid, sh, sha1sum, sha256sum,
sha3sum, sha512sum, showkey, shuf, slattach, sleep, smemcap, softlimit, sort, split, start-stop-daemon, stat,
strings, stty, su, sulogin, sum, sv, svlogd, swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail,
tar, tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top, touch, tr, traceroute, traceroute6,
true, tty, ttysize, tunctl, ubiattach, ubidetach, ubimkvol, ubirmvol, ubirsvol, ubiupdatevol, udhcpc, udhcpd,
udpsvd, umount, uname, unexpand, uniq, unix2dos, unlink, unlzma, unlzop, unxz, unzip, uptime, users, usleep,
uudecode, uuencode, vconfig, vi, vlock, volname, wall, watch, watchdog, wc, wget, which, who, whoami, whois,
xargs, xz, xzcat, yes, zcat, zcip
```

Commands available in BusyBox 1.23 (~440 commands)

Applet highlight - BusyBox vi

- If you are using BusyBox, adding vi supports only adds 20K. (built with shared libraries, using uClibc)
- You can select which exact features to compile in
- Users hardly realize that they are using a lightweight vi version!
- Tip: you can learn vi on the desktop, by running the vimtutor command



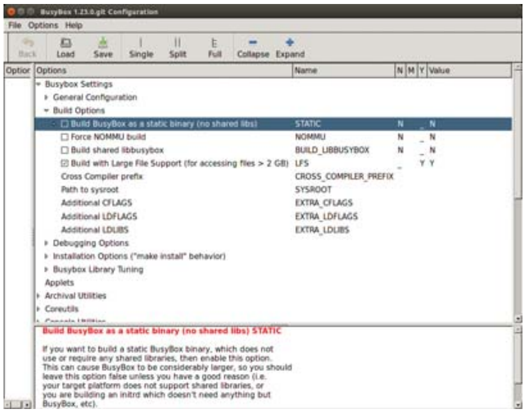
Configuring BusyBox

- Get the latest stable sources from http://busybox.net
- Configure BusyBox (creates a .config le):
  - `make defconfig`
    Good to begin with BusyBox.
    Configures BusyBox with all options for regular users.
  - `make allnoconfig`
    Unselects all options. Good to configure only what you need.
- `make gconfig (apt-get install libglade2-dev)`
  or `make menuconfig` (text)
  Same configuration interfaces as the ones used by the Linux kernel (though older versions are used).

BusyBox make xconfig

You can choose:

- the commands to compile,
- and even the command options and features that you need!



Option	Options	Name	N	M	Y	Value
	▾ Busybox Settings					
	▸ General Configuration					
	▾ Build Options					
	□ Build BusyBox as a static binary (no shared libs)	STATIC	N	_		N
	□ Force NOMMU build	NOMMU	N	_		N
	□ Build shared libbusybox	BUILD_LIBBUSYBOX	N	_		N
	☑ Build with Large File Support (for accessing files > 2 GB)	LFS	_		Y	Y
	Cross Compiler prefix	CROSS_COMPILER_PREFIX				
	Path to sysroot	SYSROOT				
	Additional CFLAGS	EXTRA_CFLAGS				
	Additional LDFLAGS	EXTRA_LDFLAGS				
	Additional LDLIBS	EXTRA_LDLIBS				
	▸ Debugging Options					
	▸ Installation Options ("make install" behavior)					
	▸ Busybox Library Tuning					
	Applets					
	▸ Archival Utilities					
	▸ Coreutils					

Build BusyBox as a static binary (no shared libs) STATIC

If you want to build a static BusyBox binary, which does not use or require any shared libraries, then enable this option. This can cause BusyBox to be considerably larger, so you should leave this option false unless you have a good reason (i.e. your target platform does not support shared libraries, or you are building an initrd which doesn't need anything but BusyBox, etc).

Compiling BusyBox

- Can compile on host or Bone
- See http://elinux.org/EBC_BusyBox

Alternative to BusyBox: embutils

http://www.fefe.de/embutils/
From the creator of diet libc

- A similar set of tiny utilities for embedded systems. Version 0.19 (Aug. 2008): 90 common commands are implemented
- Can only be built statically with diet libc!
- Compared to BusyBox: Much less momentum, user and developer base. (Last date 2008) Still misses key commands and features (ifconfig, for example)
- But can achieve smaller size than BusyBox on standalone executables

Alternatives: coreutils

- http://www.gnu.org/software/coreutils/
- Basic file, shell and text manipulation utilities of the GNU operating system
- These are the core utilities which are expected to exist on every operating system
- Previously these utilities were offered as three individual sets of GNU utilities, Fileutils, Shellutils, and Textutils.
  - Those three have been combined into a single set of utilities called Coreutils
- 14-Feb-2013