Embeded linux 101

An introduction to Embedded Linux
uclibc/busybox/buildroot

Diego Iastrubni <elcuco@kde.org>

Our new product

- We need to build a consumer device
- Needs to be a computer.
- We need a 100mhz CPU, 8mbRAM, 32MB NAND
- Must power on in 5 seconds
- Bullet proof

Option 1 - build my own OS

- Just boot into my OS
- Install grub, and create my app
- Full control
- Fast
- Demo

Problems

- I am an 1d10t, this is too much work
- Fsck this, I use linux as the kernel, more better HW support
- Grub is too complex, I want smaller
- Debian is too big
- Bash + glibc + whatever – too much

Busybox – all linux utilities in 1 exe

- All programs share code for parsing args.
- Actually much more code is shared
- We do not want to load from NAND/disk – too slow
- Dynamic linking – optional, lets statically build this

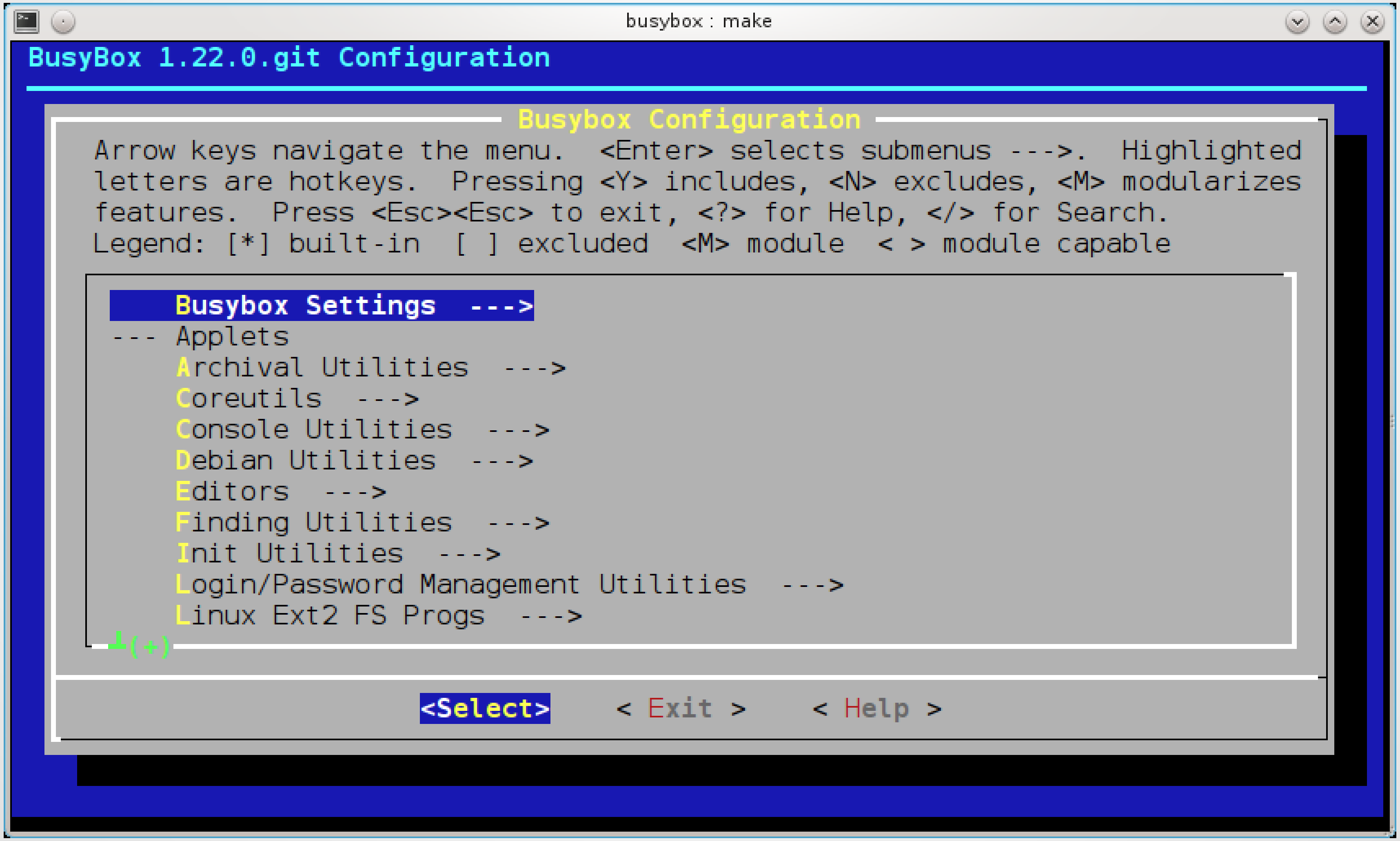
Busybox

- Busybox binary is a single app which contains many commands (sh, cat, tail, echo, vi, nc, tr, sed, ifconfig, dmesg, lsmod, insmod, fsck...)
- The binary is linked to several file names (/bin/sh → /bin/busybox, /usr/bin/vi → /bin/busybox)
- When busybox is executed without params, it checks out it's argv[0], and assumes this to be the applet to execute
- Each applet has it's own mini-main, and shares all code with the rest of the applets.

Busybox - continued

- Usually statically linked. All it needs is a linux kernel.
- The build system is the same as the linux kernel, easy to enable/disable features
- Code is easy and fun to understand
- Contains also init system, and bootchartd
- Basically – a whole “linux” userspace in a single command.
- There is an install command that creates the FSH we all know

Busybox – make menuconfig

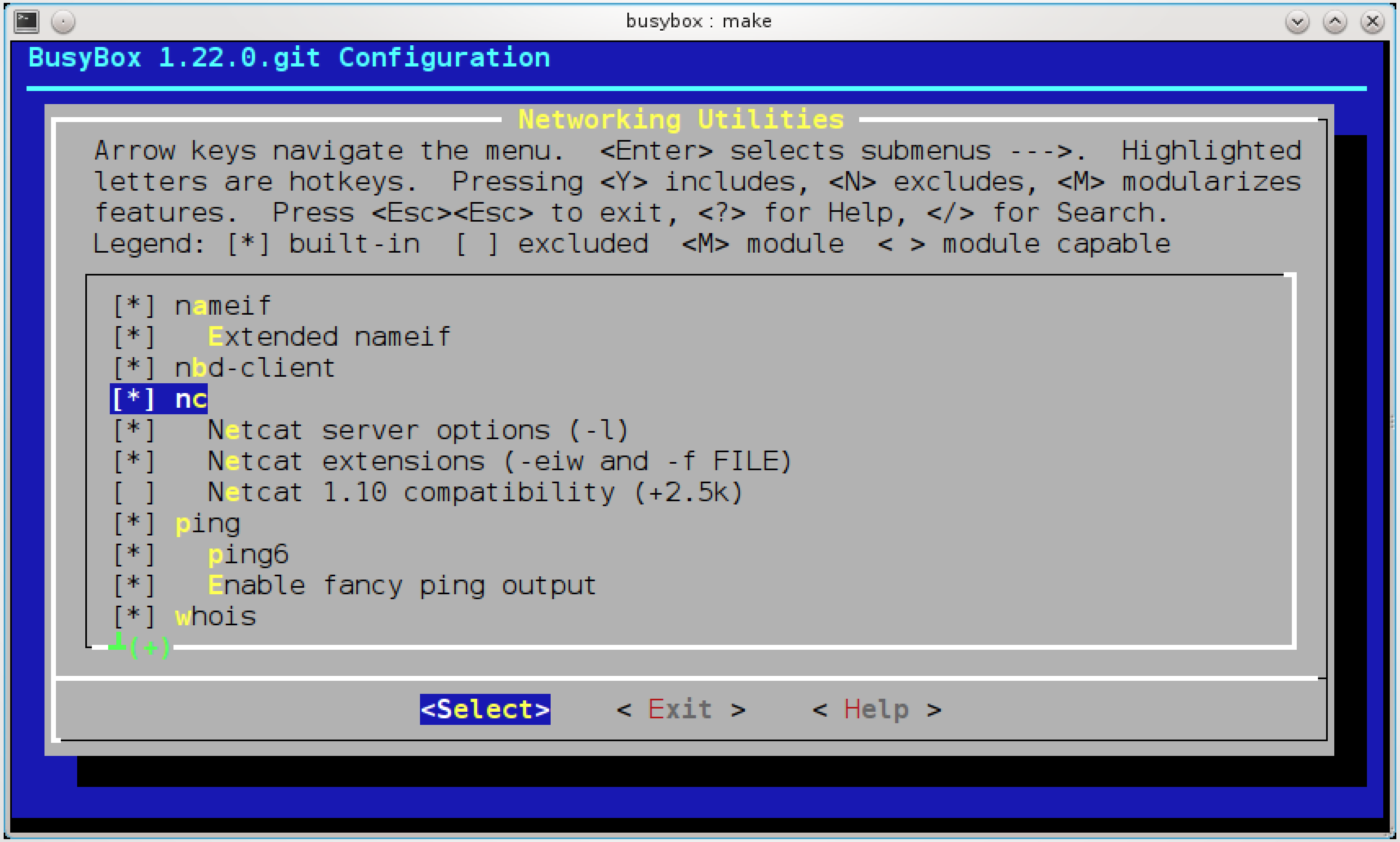


```
busybox : make
BusyBox 1.22.0.git Configuration
                         Busybox Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> 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  --->
  ↓(+)

              <Select>    < Exit >    < Help >
```

Busybox Networking Utilities menu



```
busybox : make
BusyBox 1.22.0.git Configuration
                        Networking Utilities
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

   [*] nameif
   [*]   Extended nameif
   [*] nbd-client
   [*] nc
   [*]   Netcat server options (-l)
   [*]   Netcat extensions (-eiw and -f FILE)
   [ ]   Netcat 1.10 compatibility (+2.5k)
   [*] ping
   [*]   ping6
   [*]   Enable fancy ping output
   [*] whois
   ↓(+)

          <Select>    < Exit >    < Help >
```

Demo – cross compiling

- I will demostrate how to compile Busybox for ARM.
- I will demo this on Android phone
- WAIT! Android does not use GLIBC, it uses Bionic! So what, lets statically compile!
- Anyone volunteers his phone? :-)

Busybox source - “chvt”

```
/* vi: set sw=4 ts=4: */
/*
 * Mini chvt implementation for busybox
 *
 * Copyright (C) 1999-2004 by Erik Andersen <andersen@codepoet.org>
 *
 * Licensed under GPLv2 or later, see file LICENSE in this source tree.
 */
#include "libbb.h"

int chvt_main(int argc, char **argv) MAIN_EXTERNALLY_VISIBLE;
int chvt_main(int argc UNUSED_PARAM, char **argv)
{
   int num = xatou_range(single_argv(argv), 1, 63);
   console_make_active(get_console_fd_or_die(), num);
   return EXIT_SUCCESS;
}
```

Busybox source - “clear”

```
/* vi: set sw=4 ts=4: */
/*
 * Mini clear implementation for busybox
 *
 * Copyright (C) 1999-2004 by Erik Andersen <andersen@codepoet.org>
 *
 * Licensed under GPLv2 or later, see file LICENSE in this source tree.
 */
#include "libbb.h"

int clear_main(int argc, char **argv) MAIN_EXTERNALLY_VISIBLE;
int clear_main(int argc UNUSED_PARAM, char **argv UNUSED_PARAM)
{
   /* home; clear to the end of screen */
   return full_write1_str("\033[H""\033[J") != 6;
}
```

What is a toolchain?

- Compiler – translates higher language to machine code
- Has it's own “includes” - which means it's own libraries
- By default, we link against “-libc -lm”, those are the glibc variants, bloated.

uclibc, glibc is too big

- Even if we statically link glibc,it is too big
- Lets use another toolchain, which uses a smaller libc. Lets call it micro-libc, uclibc
- Much much smaller, but getting very functional (now support dynamic loading, locales and many useless features, but can be removed!)
- We can create a toolchain (gcc) which links against uclibc

uclibc + busybox = small system

- So, if we use busybox and link statically against uclibc, we might have a very basic (but very powerful system) – with around a MB disk footprint. NICE.
- Now we only need to setup the base file system (busybox install!)
- Init.d scripts, and start coding our app!
- … what if we need more tools?

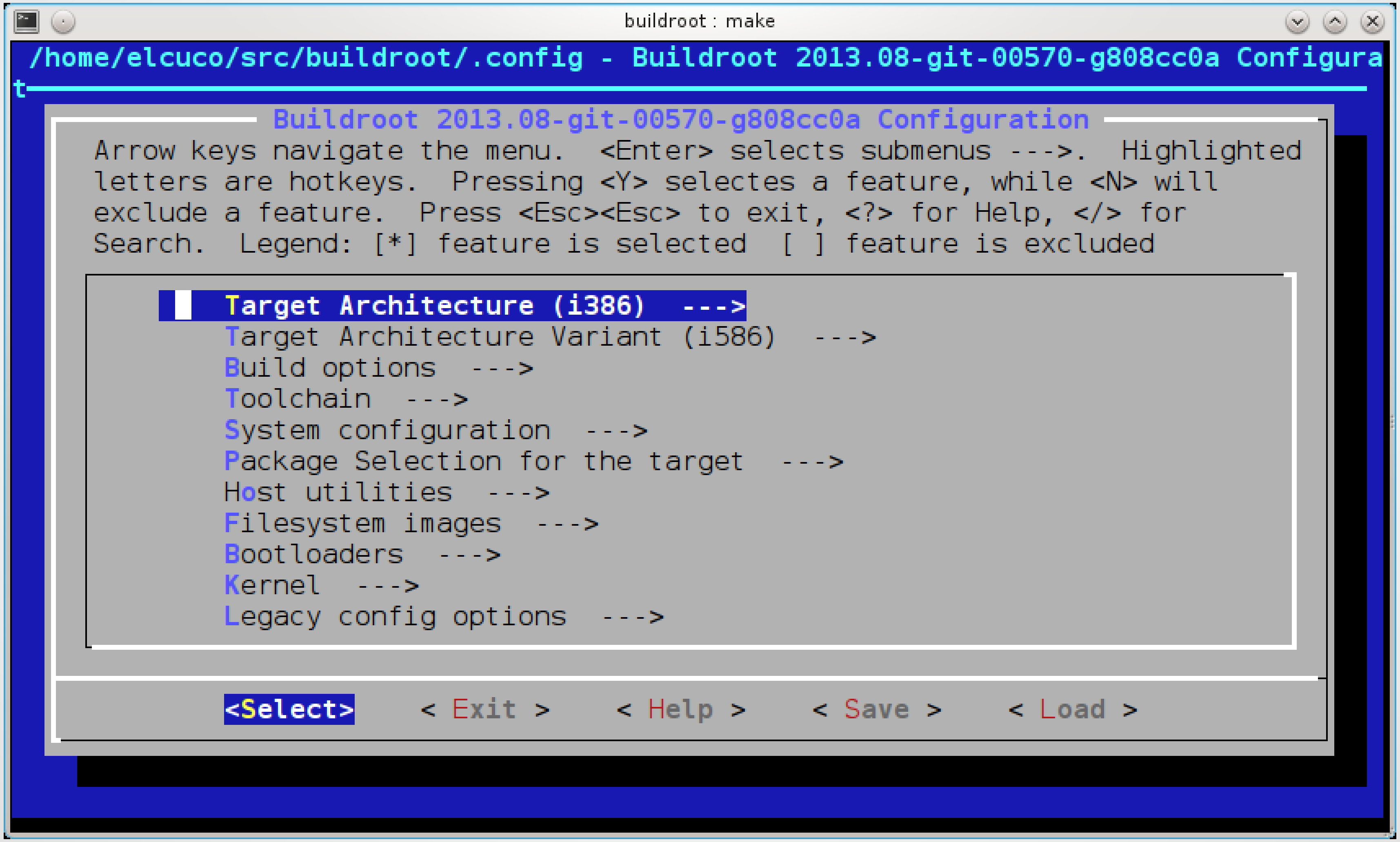
Adding new programs

- We can manually configure each program to use our toolchain, setup install prefix to our filesystem
- When we have dependencies we use -L(rootfsdir)/usr/lib
- CFLAGS=-L $(SQLITEDIR)/lib -lsqlite -L $(FOODIR)/lib -lfoo -L $(BAEDIR)/lib -lbar...
- Manually … lots of work. Lets automate.

Buildroot

- The guys that wrote uclibc found that many people had trouble using it. Creating the toolchain was hard.
- They automated the creation of the toolchain, and compilation of busybox.
- Then they added a system to download programs and compile them. Automatically.
- Example: 2 checkbox and I have X running on my FS. COOL.

Buildroot main's menu

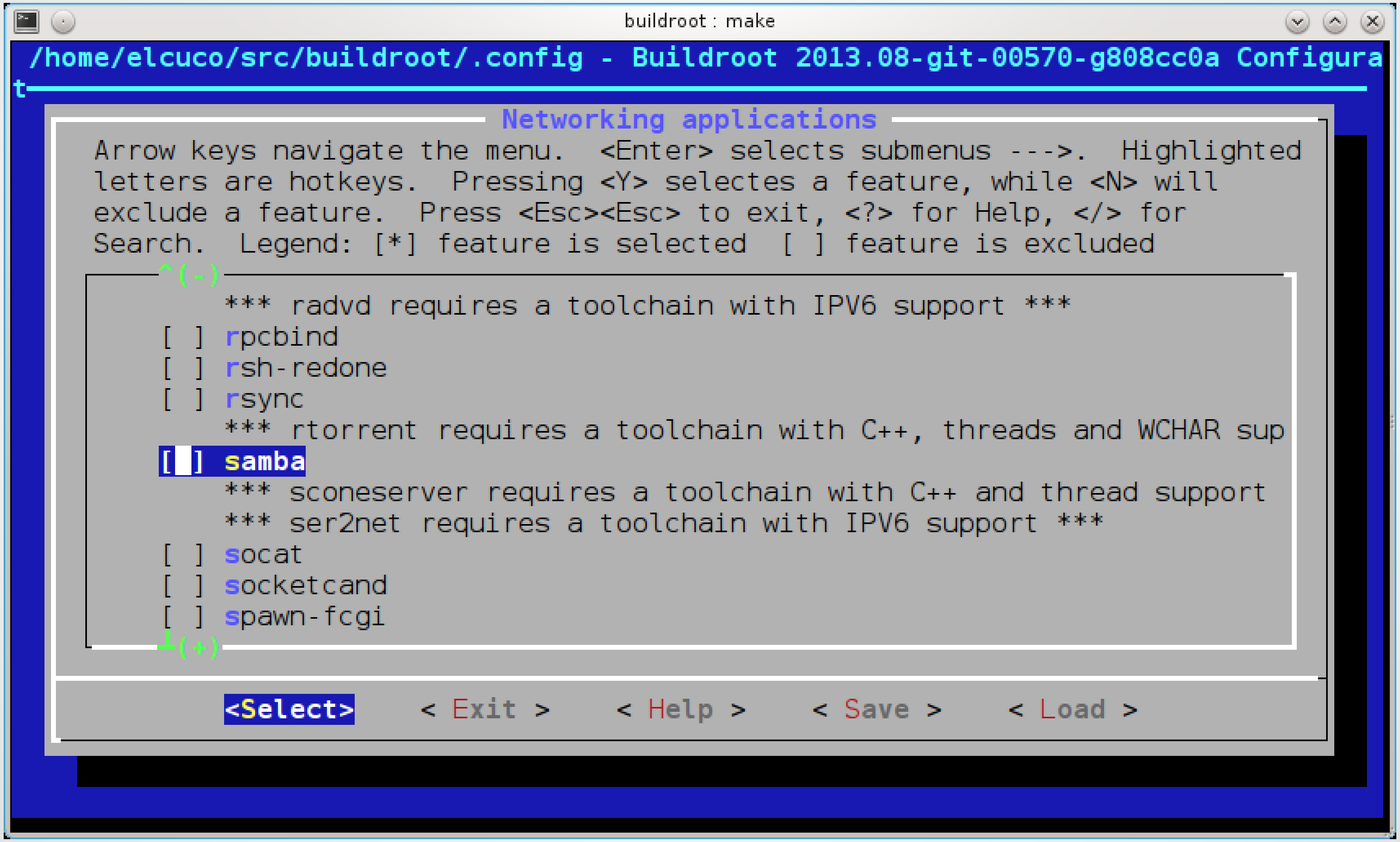


```
buildroot : make
 /home/elcuco/src/buildroot/.config - Buildroot 2013.08-git-00570-g808cc0a Configura
t
              Buildroot 2013.08-git-00570-g808cc0a Configuration
   Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
   letters are hotkeys.  Pressing <Y> selectes a feature, while <N> will
   exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </> for
   Search.  Legend: [*] feature is selected  [ ] feature is excluded

           Target Architecture (i386)  --->
           Target Architecture Variant (i586)  --->
           Build options  --->
           Toolchain  --->
           System configuration  --->
           Package Selection for the target  --->
           Host utilities  --->
           Filesystem images  --->
           Bootloaders  --->
           Kernel  --->
           Legacy config options  --->

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

Buildroot – adding Samba+rsync



```
buildroot : make
 /home/elcuco/src/buildroot/.config - Buildroot 2013.08-git-00570-g808cc0a Configura
t
                        Networking applications
    Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
    letters are hotkeys.  Pressing <Y> selectes a feature, while <N> will
    exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </> for
    Search.  Legend: [*] feature is selected  [ ] feature is excluded
          ^(-)
              *** radvd requires a toolchain with IPV6 support ***
          [ ] rpcbind
          [ ] rsh-redone
          [ ] rsync
              *** rtorrent requires a toolchain with C++, threads and WCHAR sup
          [ ] samba
              *** sconeserver requires a toolchain with C++ and thread support
              *** ser2net requires a toolchain with IPV6 support ***
          [ ] socat
          [ ] socketcand
          [ ] spawn-fcgi
          ↓(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

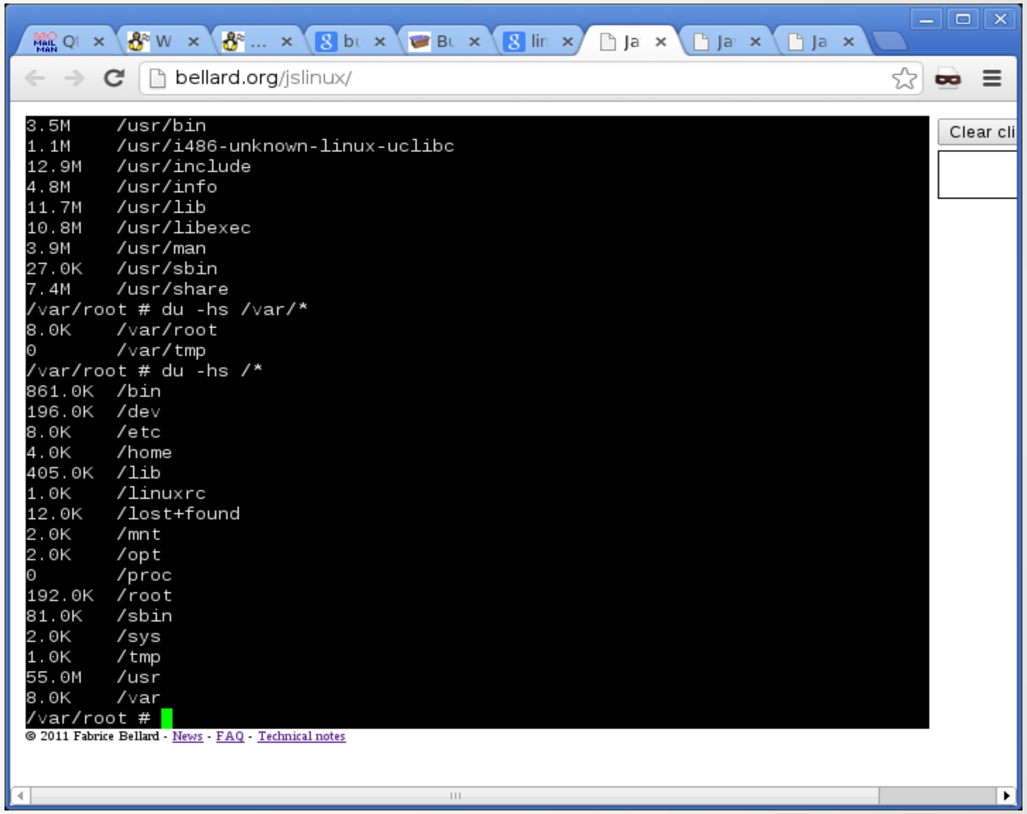
Buildroot - continued

- Buildroot is easy to setup. “git clone..” and another “make menuconfig”. Also borrowed from the Linux kernel.
- You can setup which toolchain to use
- Creates also (un)compressed images (ISO, jffs2, ext2)
- Packages added on daily basis. Project is alive.

Buildroot - alive

- How alive? Used by Fabrice the man Bellard behind ... fuck, everything! (ffmpeg, tinyc, Qemu, linux emulator in JS).
- I used it in my previous job

http://bellard.org/jslinux/



```
3.5M    /usr/bin
1.1M    /usr/i486-unknown-linux-uclibc
12.9M   /usr/include
4.8M    /usr/info
11.7M   /usr/lib
10.8M   /usr/libexec
3.9M    /usr/man
27.0K   /usr/sbin
7.4M    /usr/share
/var/root # du -hs /var/*
8.0K    /var/root
0       /var/tmp
/var/root # du -hs /*
861.0K  /bin
196.0K  /dev
8.0K    /etc
4.0K    /home
405.0K  /lib
1.0K    /linuxrc
12.0K   /lost+found
2.0K    /mnt
2.0K    /opt
0       /proc
192.0K  /root
81.0K   /sbin
2.0K    /sys
1.0K    /tmp
55.0M   /usr
8.0K    /var
/var/root #
```

© 2011 Fabrice Bellard - News - FAQ - Technical notes

Buildroot + Qt

- Buildroot has the ability to cross compile Qt
- You can easily add your App and rebuild the App
- Build natively on your PC – deploy on ARM
- Uses QVFB, quite obsolete
- No support for Qt/X11 on Buildroot
- No Wayland yet. Not even the other thign.

Problems with … traditional embedded linux

- Using “plain” linux is too bare metal.
- Each different – support is a pain.
- Missing higher level features
- Buildroot is aimed for cross compiling... native development is hard

NKOTB – Firefox OS, boot to Qt, Ubuntu Phone

- Both use the lower level Android subsystems
- Replacing Dalvik with Gecko or a Qt system.
- No release date for all 3.
- Boot to Qt just announced.

End

Questions?

Thank you

Diego Iastrubni elcuco@kde.org, diegoiast@gmail.com

Code avaialble at