CELF Embedded Linux Conference Europe

October 15 & 16, 2009

Customizing Embedded Linux Systems with PTXdist

Robert Schwebel <r.schwebel@pengutronix.de>

Pengutronix.

Agenda

- About Pengutronix - Why do we do that?
- Design Criteria for PTXdist
- How to build an embedded Linux system?
- What do Packages Do?
- Workspace Concept
- Building Cross GCCs: OSELAS.Toolchain( )
- Other cool features

Pengutronix

- Consulting, Support, Development Services for Embedded Linux

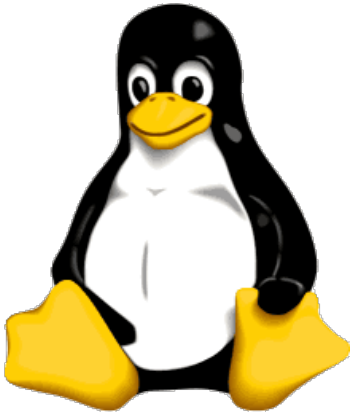

Automation & Machine Industry


Medical Devices


Automotive


Communication Industry


Energy & Oil Field Automation



Pengutronix.

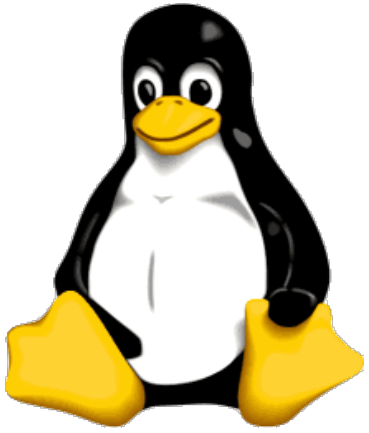
Some typical Embedded Linux systems ...

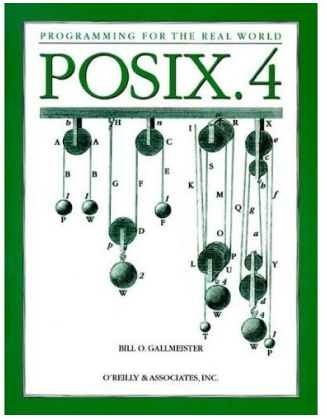
- CE and non CE projects
- The problems are the same ...

Pengutronix.

Linux & OSS are good, because ...

- ... we have control over the sources
- ... bugs can be fixed
- ... quick adaption of new features
- ... systems can be customized exactly to our needs





Pengutronix.

Some questions

- Do we have control over the sources?
  - Can we manage a patch collection against upstream? (mainlining helps, of course)
  - Can „hacks“ be maintained for a long time?
  - Can we recompile the system: ARM OABI vs. EABI, softfloat vs. hardfloat VFP, with and without debug, debug in glibc ...?
- Most standard distributions have no mechanism for these requirements!

Some questions

- Can bugs really be fixed?
  - Can we fix bugs *now*?
  - Can everything be cross compiled?
  - Care of endianess Issues?
  - Can we change configuration? (i.e. quickboot kernel .config)
  - Do we immediately get new software versions once we have mainlined our patches?

Some questions

- Can we really adapt new features quickly?
  - Does our laser controller need the same features as Ubuntu?
  - Do we get new features when our *project* demands them?
  - Can we stay with selected old versions if we want to?
    (i.e. gtk -> broken with newer DirectFB backends)

Some questions

- Can systems be customized exactly to our needs?
- Customizing is the most important feature for us!
- Examples:
  - Flicker-free booting of 400 MHz i.MX27 into Qt in < 6 s
  - Small headless systems with something like 8 MB RAM
  - We don't have SQL databases, Perl, ... but distros often rely on that
  - Adapt kernel + userland to well-know embedded hardware
    No need for initramfs, module loading etc.

Have Systems to be small?

- Size doesn't matter that much any more, these days:

  phyCORE-i.MX35
  1 GB NAND flash

- We still have systems with 16 MB NOR flash in the field!
- Most Standard distros can't be scaled below about 300 MB without losing functionality (in-field package update)
- NAND is good for space requirements
- Reliability...?

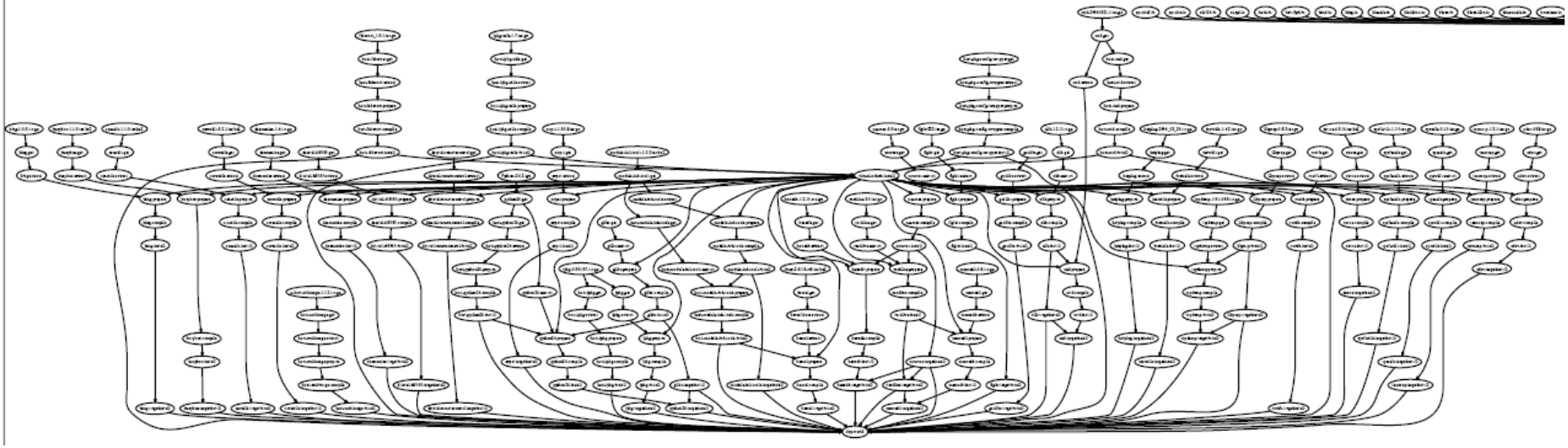
Pengutronix.

Entering PTXdist: Build your own Linux

- Most standard distros don't fit our needs (at least not at the moment)
  - They cannot be reproduced quickly enough.
  - Too many packet inter-dependencies which don't matter on embedded systems
  - Customizing is a big problem.
- So for embedded usage, our current policy is „build your own“.
- This may change in the future (customize moblin with ptxdist?)

How do you build an Embedded Linux?

- Just do this, in the right order:



- And this is only a headless realtime system,

  without gtk+glib+atk+pango+cairo+...,
  without dbus,
  without Qt,
  without x.org, ...

Pengutronix.

PTXdist: Building Blocks

- What do we have to do, for the whole system?

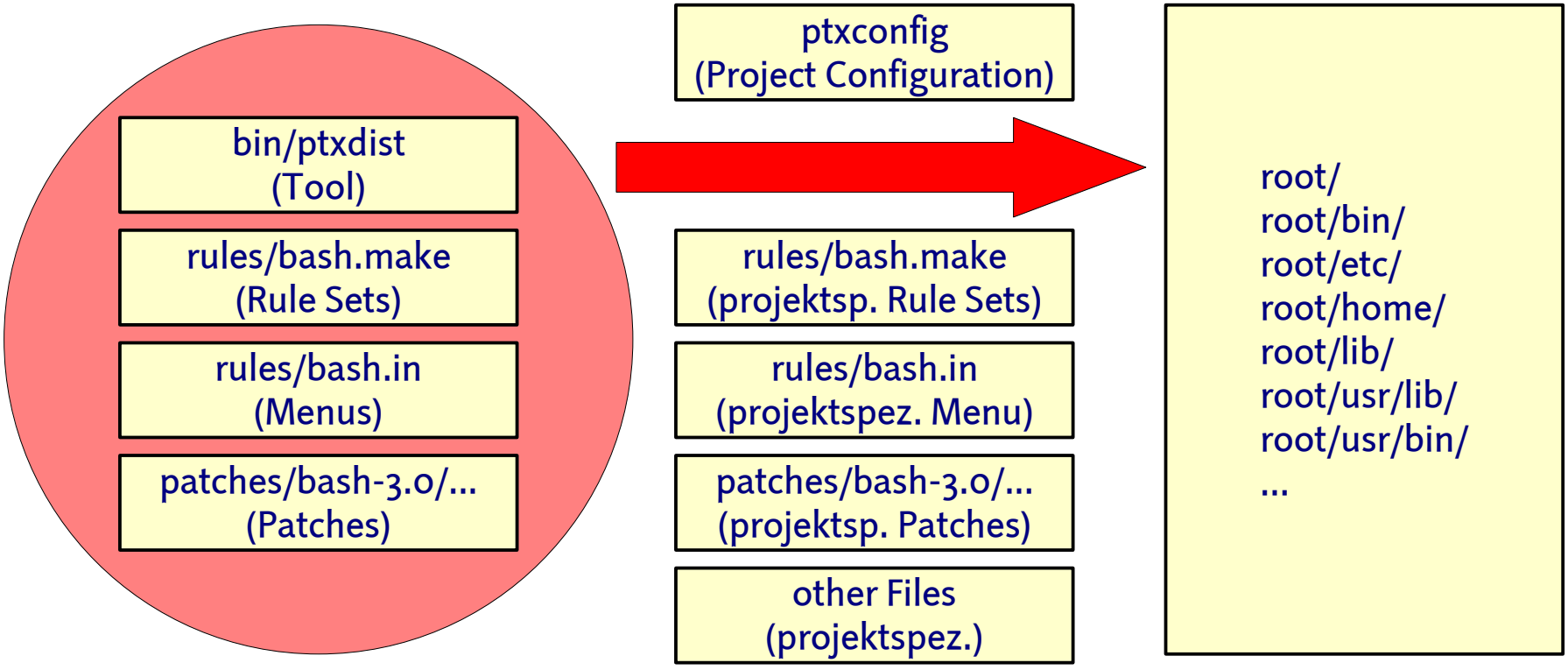
  1) Configure which packages to have on the target.

  2) Configure options for the packages.

  3) „ptxdist go“ -> Do All Necessary Things (TM)

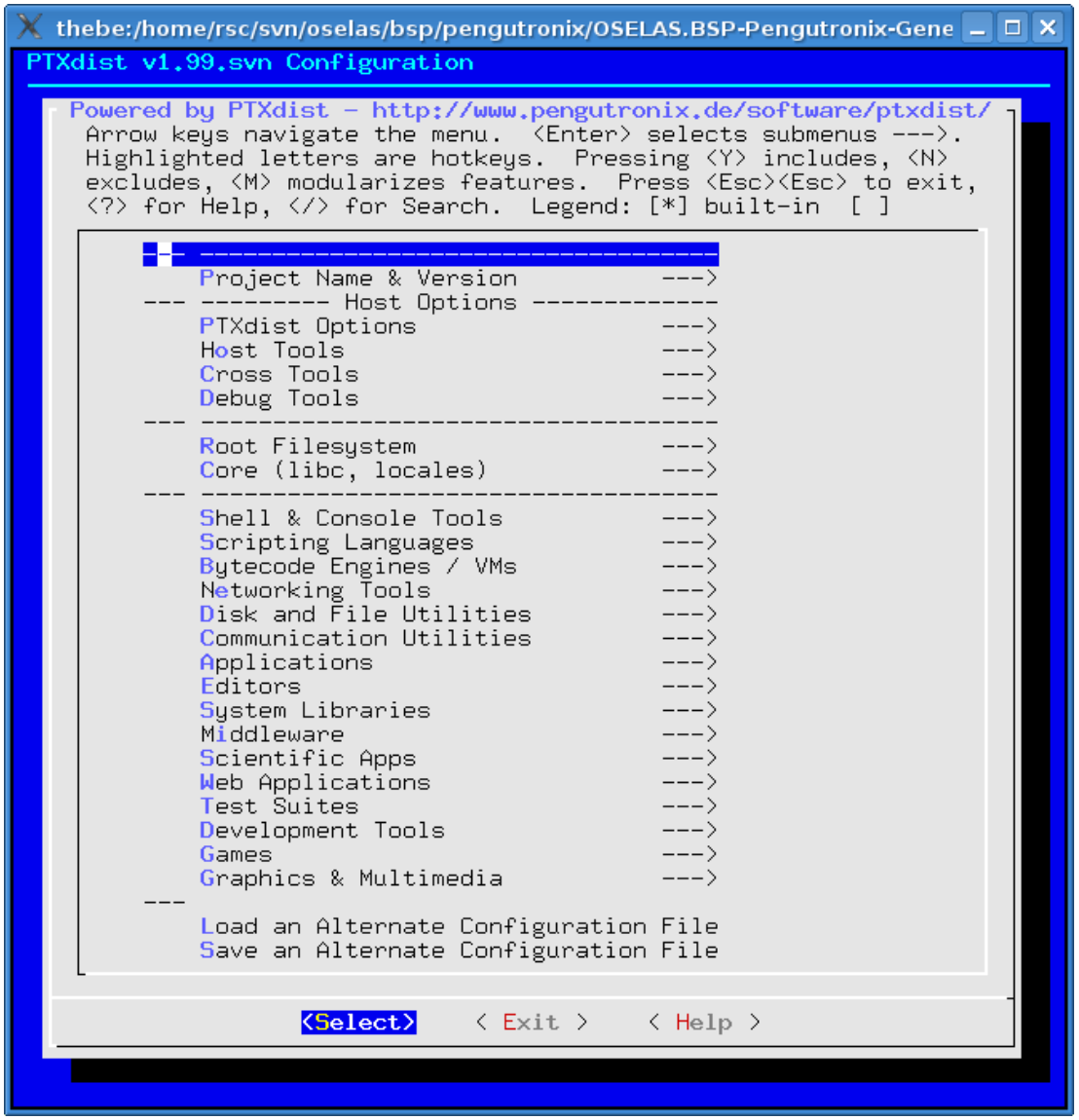
  4) Find out all dependencies and kick stages in right order.

Packages can be overwritten on workspace:



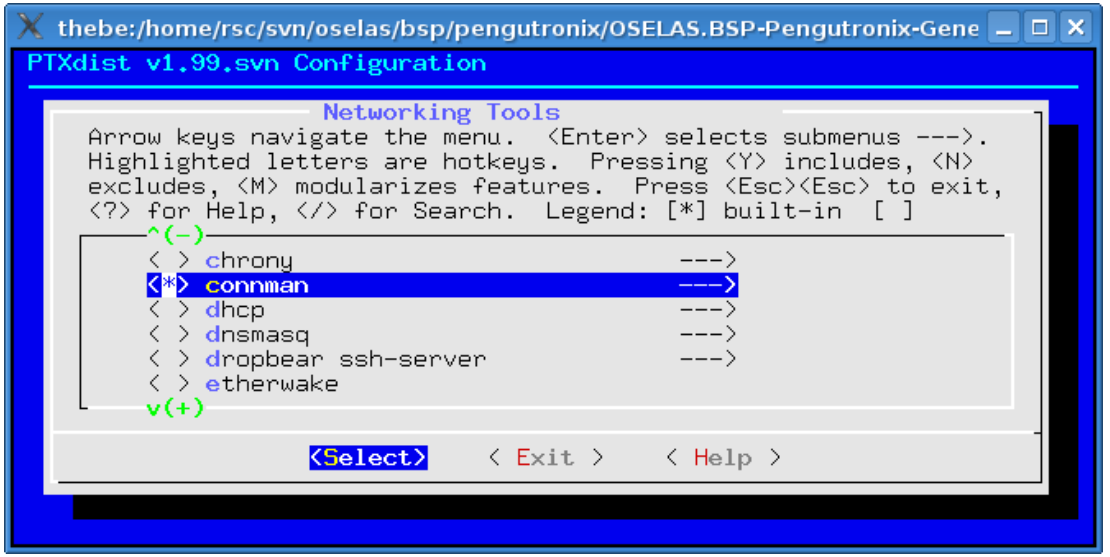
Configuration

- Kconfig based - kernel hacker compatible mouse-less operation

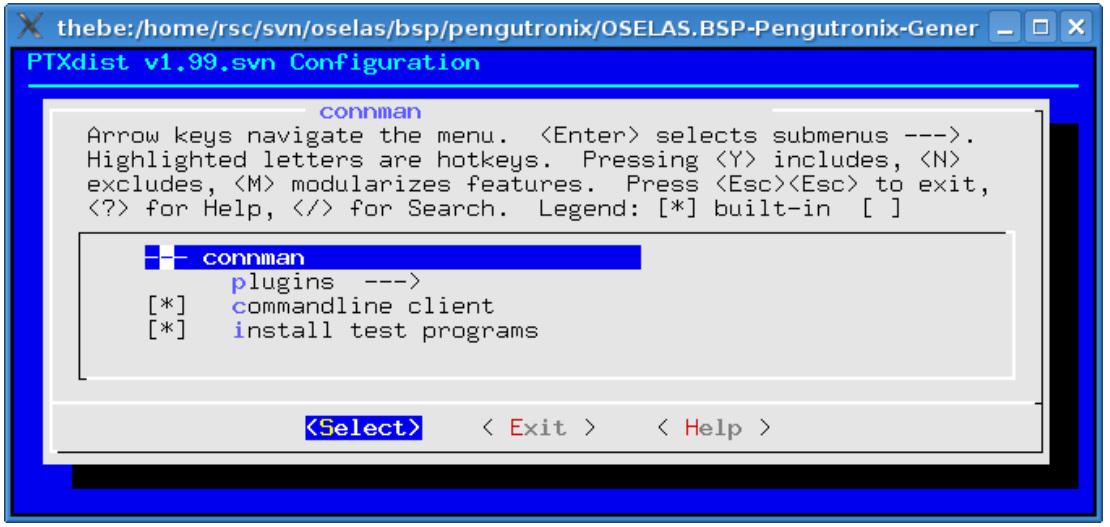


Configuration

- Which Packages will go into our distribution?



- How are the package configured?



Pengutronix.

Configuration

- The result of the configuration is a .config style file:
- valid shell syntax valid make syntax
- The configuration contains both information:

  - what to build
  - how to build it

selected_ptxconfig + (~/svn/...engutronix-Generic-trunk) - VIM

```
#
# Networking Tools
#
# PTXCONF_APACHE2 is not set
# PTXCONF_APACHE2_MOD_PYTHON is not set
# PTXCONF_BETAFTPD is not set
# PTXCONF_BIND is not set
# PTXCONF_BING is not set
# PTXCONF_CANFESTIVAL is not set
# PTXCONF_CHRONY is not set
PTXCONF_CONNMAN=y

#
# plugins
#
# PTXCONF_CONNMAN_LOOPBACK is not set
PTXCONF_CONNMAN_ETHERNET=y
# PTXCONF_CONNMAN_WIFI is not set
# PTXCONF_CONNMAN_WIMAX is not set
# PTXCONF_CONNMAN_BLUETOOTH is not set
# PTXCONF_CONNMAN_UDHCP is not set
# PTXCONF_CONNMAN_DHCLIENT is not set
# PTXCONF_CONNMAN_RESOLVCONF is not set
# PTXCONF_CONNMAN_DNSPROXY is not set
# PTXCONF_CONNMAN_HSO is not set
# PTXCONF_CONNMAN_UDEV is not set
# PTXCONF_CONNMAN_POLKIT is not set
# PTXCONF_CONNMAN_FAKE is not set
PTXCONF_CONNMAN_CLIENT=y
PTXCONF_CONNMAN_TESTS=y
```

Pengutronix.

Liftoff

- „ptxdist go“

  Do all Necessary Things (TM)

- Execute stages:

  get
  extract
  prepare
  compile
  install
  targetinstall

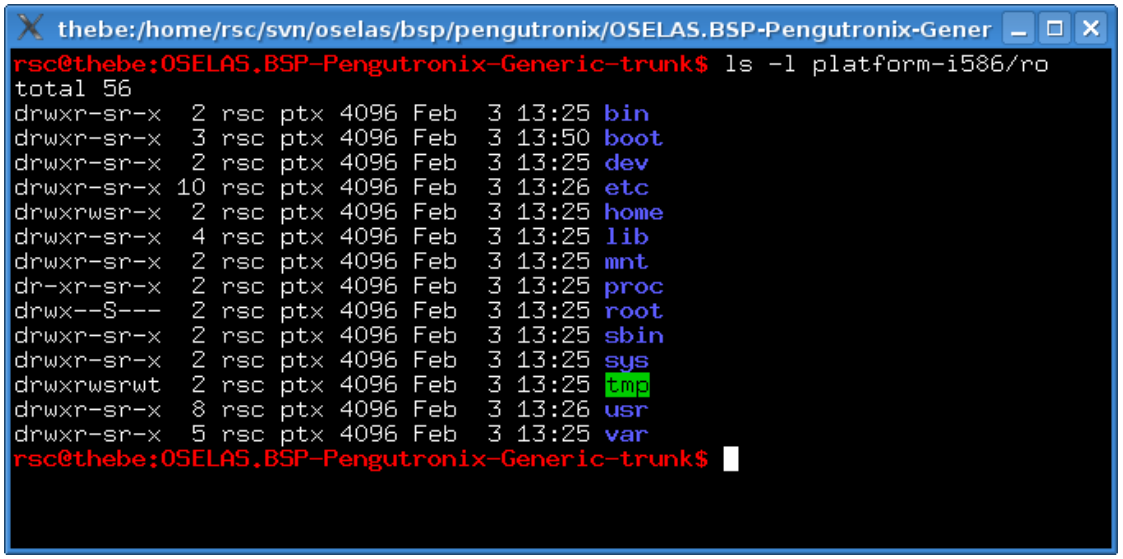
- Solve Dependencies

```
thebe:/home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-Gener
rsc@thebe:OSELAS.BSP-Pengutronix-Generic-trunk$ ./p go

------------------
target: connman.get
------------------

Finished target connman.get

-----------------------
target: connman.extract
-----------------------

extract: archive=/ptx/src/connman-0.10.tar.gz
extract: dest=/home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-
Generic-trunk/platform-i586/build-target
PATCHIN: packet=connman-0.10
PATCHIN: dir=/home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-G
eneric-trunk/platform-i586/build-target/connman-0.10
PATCHIN: no patches for connman-0.10 available
Fixing up /home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-Gene
ric-trunk/platform-i586/build-target/connman-0.10/configure
Fixing up /home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-Gene
ric-trunk/platform-i586/build-target/connman-0.10/ltmain.sh
Finished target connman.extract

-----------------------
target: connman.prepare
-----------------------

cd /home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-Generic-tru
nk/platform-i586/build-target/connman-0.10 && \
               PATH=/home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengu
tronix-Generic-trunk/platform-i586/sysroot-cross/bin:/home/rsc/svn/oselas/
bsp/pengutronix/OSELAS.BSP-Pengutronix-Generic-trunk/platform-i586/sysroot
-cross/sbin:/home/rsc/svn/oselas/bsp/pengutronix/OSELAS.BSP-Pengutronix-Ge
neric-trunk/platform-i586/sysroot-cross/lib/udev:$PATH AR=i586-unknown-lin
ux-gnu-ar AS=i586-unknown-linux-gnu-as LD=i586-unknown-linux-gnu-ld NM=i58
6-unknown-linux-gnu-nm CC=i586-unknown-linux-gnu-gcc CXX=i586-unknown-linu
x-gnu-g++ CPP=i586-unknown-linux-gnu-cpp RANLIB=i586-unknown-linux-gnu-ran
lib READELF=i586-unknown-linux-gnu-readelf OBJCOPY=i586-unknown-linux-gnu-
objcopy OBJDUMP=i586-unknown-linux-gnu-objdump STRIP=i586-unknown-linux-gn
u-strip DLLTOOL=i586-unknown-linux-gnu-dlltool CC_FOR_BUILD=gcc CPP_FOR_BU
ILD=gcc LINK_FOR_BUILD=gcc  CPPFLAGS='-isystem /home/rsc/svn/oselas/bsp/pe
```

The Result of „ptxdist go“

- build root filesystem
  in the platform dir
- „root/“ is
  NFS mountable
- „root-debug“:
  for gdbserver use
- Development workflow:

  - boot kernel via TFTP
  - mount root/ with NFS-root.



What do packages do?

- A „package“ consists of:
  - configuration, by a Kconfig file „packagename.in“
  - a rule set, specified in „packagename.make“
  - maybe a quilt stack of patches
- The package header contains definitions:

connman.make (~/svn/ptxdist-trunk-clean/rules) - VIM

```
#
# Paths and names
#
CONNMAN_VERSION  := 0.10
CONNMAN          := connman-$(CONNMAN_VERSION)
CONNMAN_SUFFIX   := tar.gz
CONNMAN_URL      := http://ftp.moblin.org/connman/releases/$(CONNMAN).$(CONNMAN_S
UFFIX)
CONNMAN_SOURCE   := $(SRCDIR)/$(CONNMAN).$(CONNMAN_SUFFIX)
CONNMAN_DIR      := $(BUILDDIR)/$(CONNMAN)
```

Pengutronix.

Packages

- The rest of the packagename.make file contains „stages“:

  „get“

  „extract“

  „prepare“

  „compile“

  „install“

  „targetinstall“

  „clean“

Get Stage

- Get tarball from upstream, if not already there (with fallback URL)
- Accumulate tarballs in a dir:
  can be shared for different developers

connman.make (~/svn/ptxdist-trunk-clean/rules) - VIM

```
#
# Paths and names
#
CONNMAN_VERSION := 0.10
CONNMAN         := connman-$(CONNMAN_VERSION)
CONNMAN_SUFFIX  := tar.gz
CONNMAN_URL     := http://ftp.moblin.org/connman/releases/$(CONNMAN).$(CONNMAN_S
UFFIX)
CONNMAN_SOURCE  := $(SRCDIR)/$(CONNMAN).$(CONNMAN_SUFFIX)
CONNMAN_DIR     := $(BUILDDIR)/$(CONNMAN)

# ----------------------------------------------------------------------------
# Get
# ----------------------------------------------------------------------------

$(CONNMAN_SOURCE):
        @$(call targetinfo)
        @$(call get, CONNMAN)
```

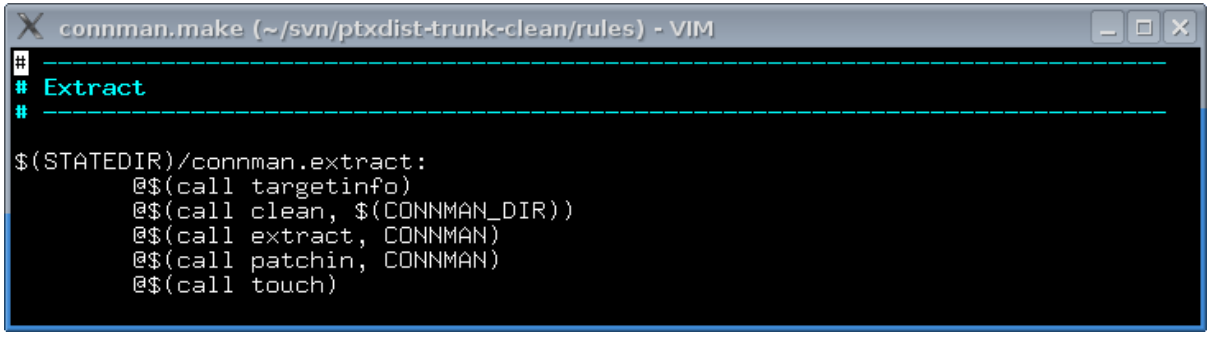
Pengutronix.

Extract Stage

- „tar xf package-x.y.z.tar.bz2“

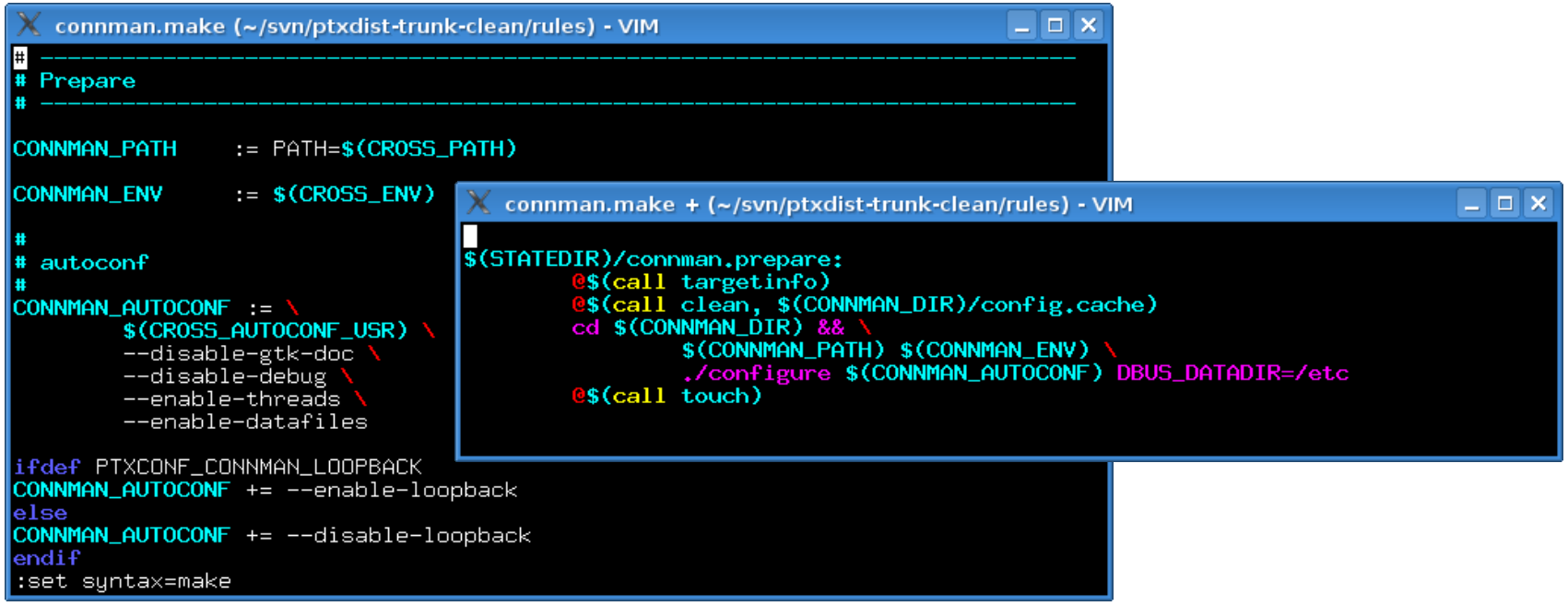
  (tar.bz2, tar.gz, zip)

- Apply patches (quilt stack)
- Fixup ltmain.sh and configure scripts to avoid path hardcoding



Prepare Stage

- „./configure --host=... --build=... --enable-foo --with-bar=baz“
- configure switches can be set in correspondence with menu entries
- non autotoolized packets :-/



Compile Stage

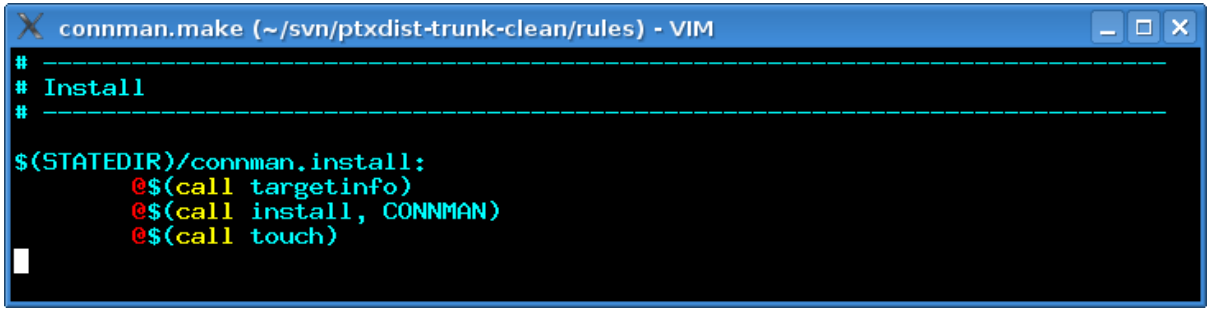
- „make“
- Multi core usage: „make -j <2*cores>“
- Broken packages are built with -j 1

connman.make (~/svn/ptxdist-trunk-clean/rules) - VIM

```
# ----------------------------------------------------------------------------
# Compile
# ----------------------------------------------------------------------------

$(STATEDIR)/connman.compile:
        @$(call targetinfo)
        cd $(CONNMAN_DIR) && $(CONNMAN_PATH) $(MAKE) $(PARALLELMFLAGS)
        @$(call touch)
```

Install Stage

- „make install DESTDIR=<somewhere>“
- development host side installation for
  libs, binaries, headers, man pages, .pc files ...



Targetinstall Stage

- „make install“: good for development, too large for the target!
- Targetinstall: full control over what goes into the image
- Package content may be dependend on menu/configuration! (different to standard distros)
- This is where the target customization takes place.

Targetinstall Stage

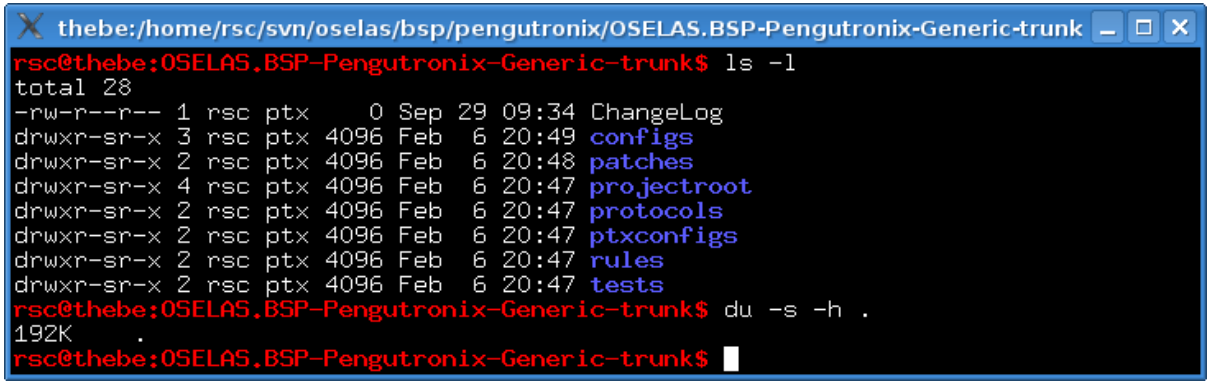
connman.make + (~/svn/ptxdist-trunk-clean/rules) - VIM

```
# ----------------------------------------------------------------------------
# Target-Install
# ----------------------------------------------------------------------------

$(STATEDIR)/connman.targetinstall:
        @$(call targetinfo)

        @$(call install_init,  connman)
        @$(call install_fixup, connman,PACKAGE,connman)
        @$(call install_fixup, connman,PRIORITY,optional)
        @$(call install_fixup, connman,VERSION,$(CONNMAN_VERSION))
        @$(call install_fixup, connman,SECTION,base)
        @$(call install_fixup, connman,AUTHOR,"Robert Schwebel <r.schwebel@pengutronix.de>")
        @$(call install_fixup, connman,DEPENDS,)
        @$(call install_fixup, connman,DESCRIPTION,missing)

        # binary
        @$(call install_copy, connman, 0, 0, 0755, -, /usr/sbin/connmand)

        # dirs
        @$(call install_copy, connman, 0, 0, 0755, /usr/lib/connman)
        @$(call install_copy, connman, 0, 0, 0755, /usr/lib/connman/scripts)
        @$(call install_copy, connman, 0, 0, 0755, /usr/lib/connman/plugins)

        # start script
        @$(call install_copy, connman, 0, 0, 0755, \
                $(PTXDIST_TOPDIR)/generic/etc/init.d/connman, \
                /etc/init.d/connman)

        # dbus config
        @$(call install_copy, connman, 0, 0, 0644, -, /etc/dbus-1/system.d/connman.conf)

        #
        # plugins
        #
ifdef PTXCONF_CONNMAN_ETHERNET
        @$(call install_copy, connman, 0, 0, 0644, -, /usr/lib/connman/plugins/ethernet.so)
endif
        @$(call install_finish, connman)

        @$(call touch)
```

Pengutronix.

Workspace Concept

- All project work is being executed on a „project workspace“, which is a directory containing all project relevant files.
- The workspace is pretty small:



Building Cross GCCs: OSELAS.Toolchain ( )

- An important prerequisite for building embedded systems are Toolchains: gcc / binutils / glibc / kernel headers
- But where to get recent toolchains from?
- Started with Dan Kegel's crosstool,
  which was an excellent choice at that time.
- We noticed that Dan had different aims than we have:
  - We want to have recent gcc + glibc versions
  - Patches have to be separated in a clean way, per tool revision
  - Problems have to be sorted out with upstream
  - Crude hacks which have been necessary in the gcc-2.95.3 era are not needed for gcc-4.3.3 any more :-)

Pengutronix.

Building Cross GCCs: OSELAS.Toolchain ( )

- We tried to be a good OSS citisen and make crosstool better, instead of forking our own project. Unfortunately, it turned out that crosstool was so broken internally, that starting from scratch was faster and cleaner.
- OSELAS.Toolchain( ) is based on ptxdist's make mechanics to deduce the order of things which have to be done
- Selecting a set of config options is a simple matter of selecting the right ptxconfig file (we have > 25 toolchains in 1.99.2).
- Building a toolchain goes like „ptxdist go“
- Patches are documented in a clean way, canonical patch headers
- Several topics have been resolved in the GCC bugzilla so far

Other Cool Features

- ipkg: Installing packages on the target is possible.
- Build system could be changed towards other packet formats, so we are neither fixed to „do-it-yourself“, nor to .ipkg
- Complete recompilation with synced config: Want to oprofile your system? Just build it completely with debug symbols ...
- Platform Abstractions: Hardware config is separated from software config; so a project specified in a ptxconfig file can be built for different hardware platforms.
- Simulation: build against KVM for development

crossdev@send-patches.org

- The idea came up on FOSDEM 2009:
- Have common mailing list for all cross-build-system people
- collect patches worth to be upstreamed
- review & make ready for prime time
- submit things to the upstream maintainers
- Help us making Linux better!

Pengutronix.

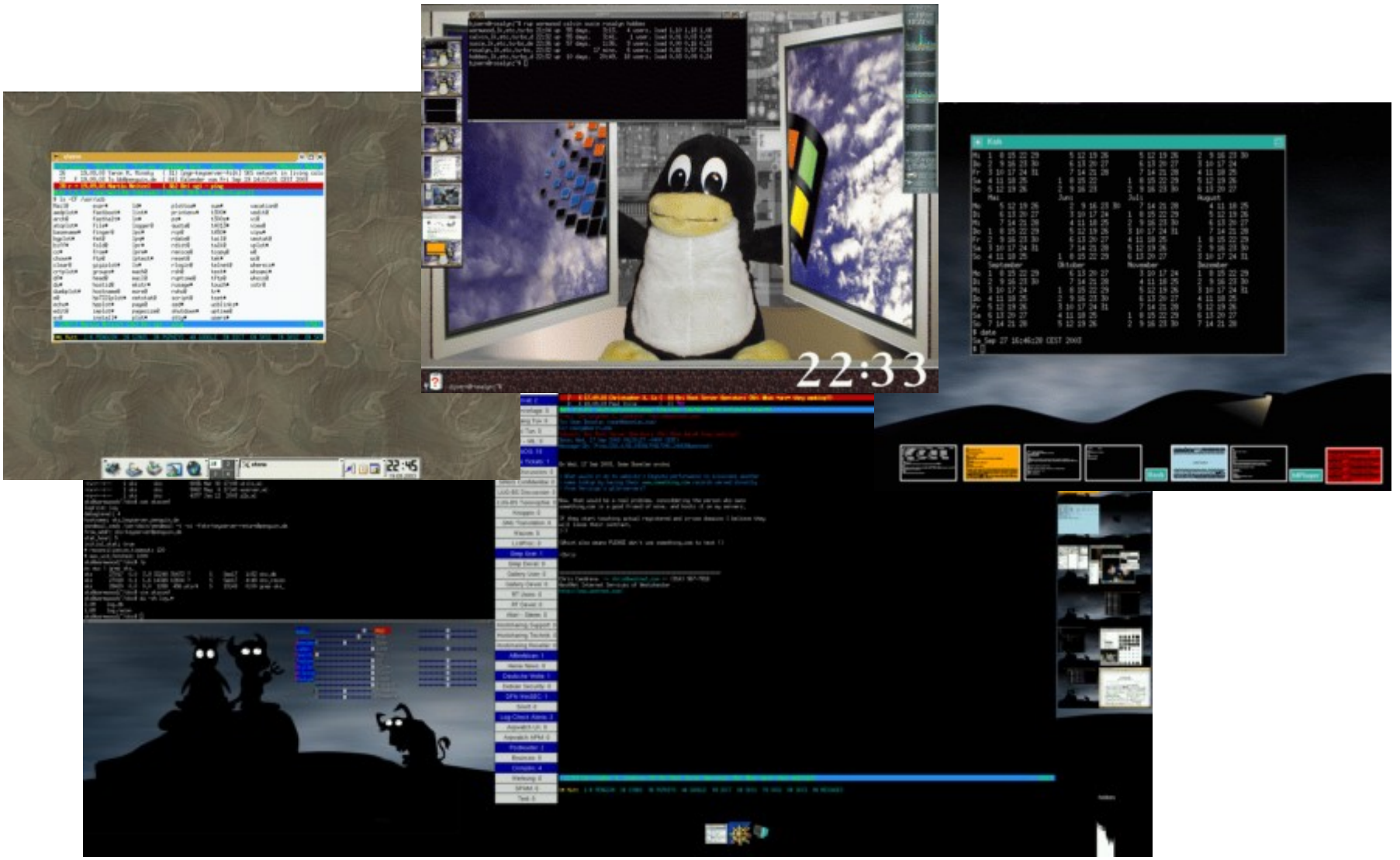
Future

- ptxdist 2.0 ...?
- Time based releases?
- Customize other things than rootfs+toolchain?

Ressources

- PTXdist Web Site:
  http://www.pengutronix.de/software/ptxdist/index_en.html
- Mailing Lists (ptxdist + send-patches.org):
  http://www.pengutronix.de/mailinglists/index_en.html
- IRC Channel:
  irc.freenode.net
  #ptxdist

Thanks for your Interest! Questions...?



Pengutronix.