From One Architecture to Many: Porting OpenMandriva to AArch64, armv7hnl, RISC-V and Ryzen

OSS/ELC 2018
Bernhard “Bero” Rosenkränzer

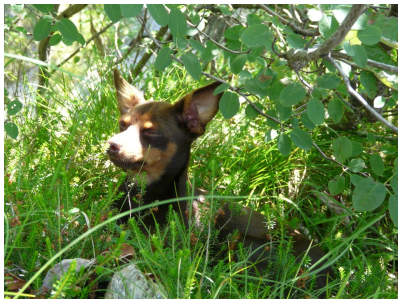
OpenMandriva

Linaro

Who are we?

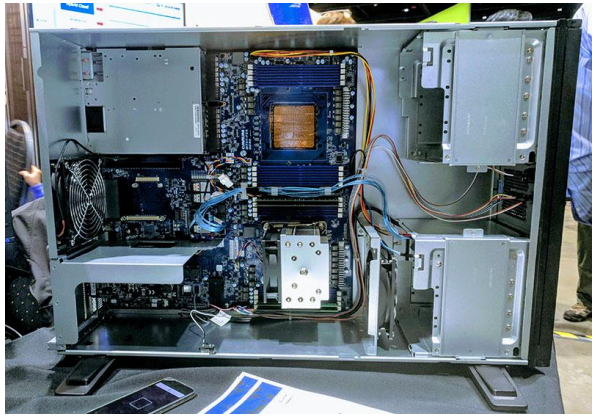
- One of the older Linux distributions still alive - started in 1998 as Mandrake, renamed to Mandriva after merging with Conectiva, renamed and reorganized as OpenMandriva when Mandriva (the company) went out of business, but “Mandriva” (the OS community) remained active
- Always primarily (though not exclusively) a desktop OS -- which up until recently meant x86.
- The upcoming version 4.0 will be released for x86, aarch64, armv7hnl and Ryzen (special variant of x86), and a RISC-V port is underway.

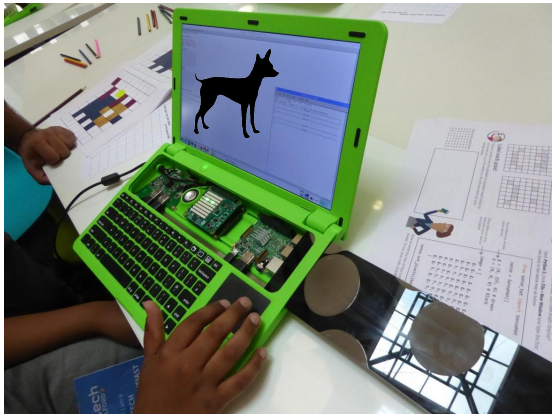
our mascot Chwido -- insisting to participate even though he couldn't make the travel



Why do this now?

- Other CPU architectures are starting to be fast enough to replace a traditional desktop and laptop
- Monopolies are harmful



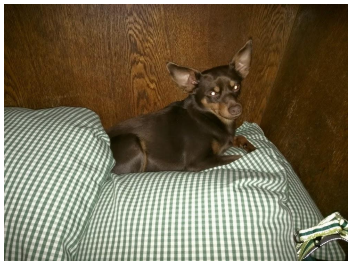


Will this be yet another port that will be discontinued?

No: because this time we're doing it right…

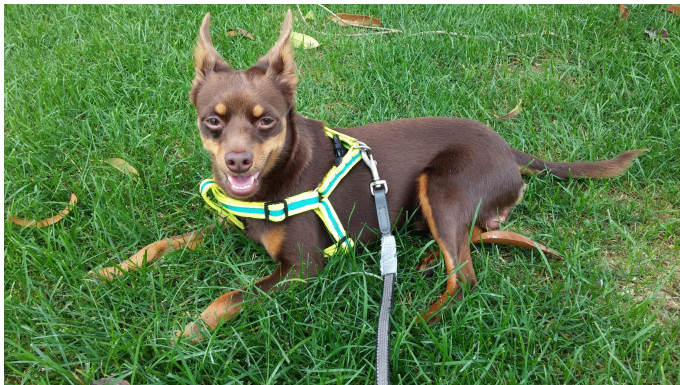
Mistakes made in previous ports that we're not going to repeat (and neither should you…):

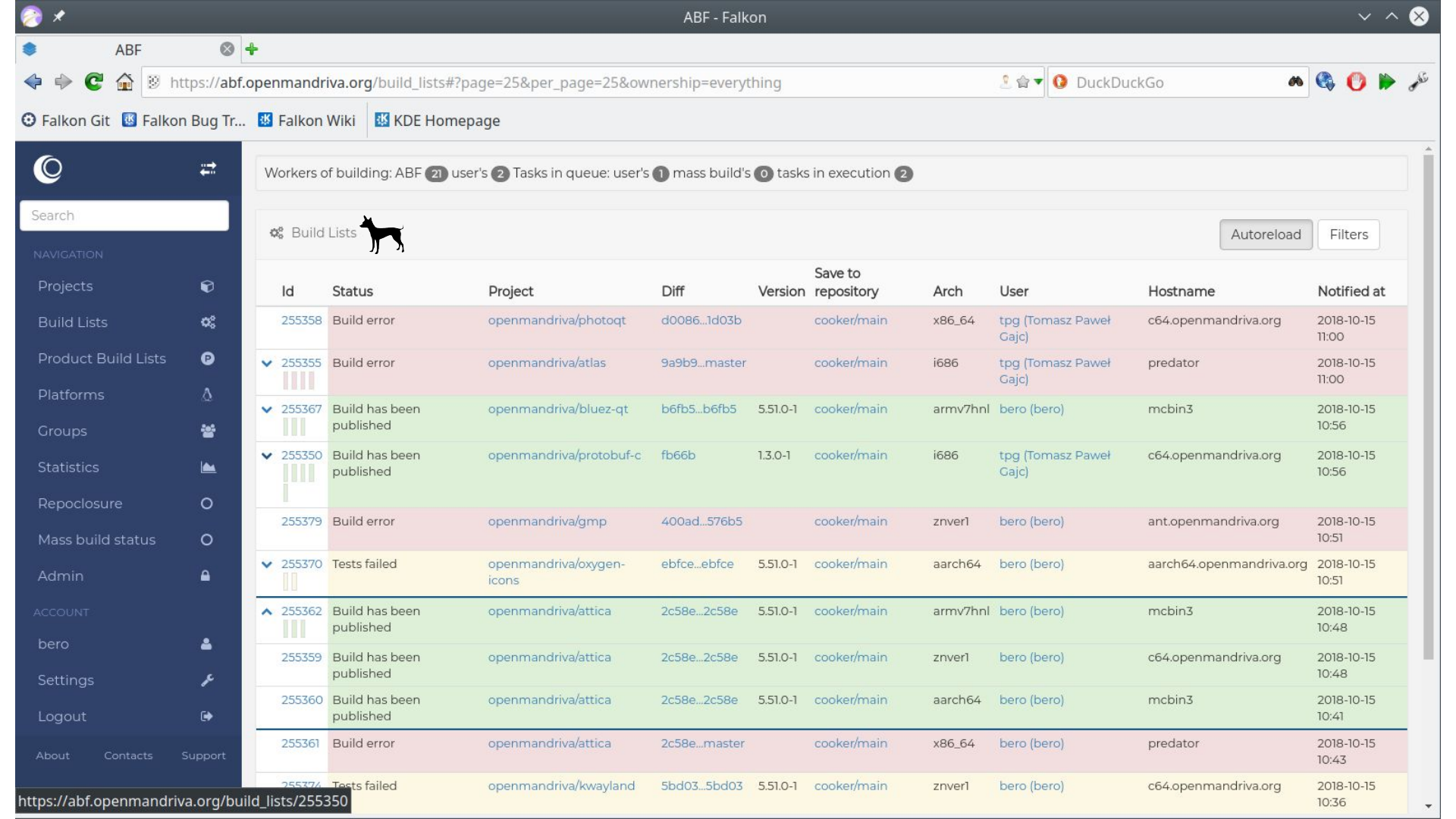
- Ports done by a company without community participation
- Porting an old release instead of porting the current development branch
- Separate toolchains for every architecture, separate crosscompilers essentially inviting architectures that get less attention to fall behind



Setting up a port the right way

- Get the toolchains including crosscompilers right
  - More on that a bit later
- Build core packages needed to compile others.
  - No shortcuts there… Still some work involved there, will come back to that
- When the core packages are ready, add it to the build system (I don’t presume anyone still uploads locally built binary packages directly…) - but allow packages to fail for the time being (it’ll take some time before all libraries are ready…)
- Try to build all packages
- Stop allowing packages that don’t build on all architectures, build install images




ABF - Falkon
ABF
https://abf.openmandriva.org/build_lists#?page=25&per_page=25&ownership=everything
DuckDuckGo
Falkon Git
Falkon Bug Tr...
Falkon Wiki
KDE Homepage
Search
NAVIGATION
Projects
Build Lists
Product Build Lists
Platforms
Groups
Statistics
Repoclosure
Mass build status
Admin
ACCOUNT
bero
Settings
Logout
About
Contacts
Support
Workers of building: ABF 21 user's 2 Tasks in queue: user's 1 mass build's 0 tasks in execution 2
Build Lists
Autoreload
Filters
Id
Status
Project
Diff
Version
Save to repository
Arch
User
Hostname
Notified at
255358 Build error openmandriva/photoqt d0086...1d03b cooker/main x86_64 tpg (Tomasz Paweł Gajc) c64.openmandriva.org 2018-10-15 11:00
255355 Build error openmandriva/atlas 9a9b9...master cooker/main i686 tpg (Tomasz Paweł Gajc) predator 2018-10-15 11:00
255367 Build has been published openmandriva/bluez-qt b6fb5...b6fb5 5.51.0-1 cooker/main armv7hnl bero (bero) mcbin3 2018-10-15 10:56
255350 Build has been published openmandriva/protobuf-c fb66b 1.3.0-1 cooker/main i686 tpg (Tomasz Paweł Gajc) c64.openmandriva.org 2018-10-15 10:56
255379 Build error openmandriva/gmp 400ad...576b5 cooker/main znver1 bero (bero) ant.openmandriva.org 2018-10-15 10:51
255370 Tests failed openmandriva/oxygen-icons ebfce...ebfce 5.51.0-1 cooker/main aarch64 bero (bero) aarch64.openmandriva.org 2018-10-15 10:51
255362 Build has been published openmandriva/attica 2c58e...2c58e 5.51.0-1 cooker/main armv7hnl bero (bero) mcbin3 2018-10-15 10:48
255359 Build has been published openmandriva/attica 2c58e...2c58e 5.51.0-1 cooker/main znver1 bero (bero) c64.openmandriva.org 2018-10-15 10:48
255360 Build has been published openmandriva/attica 2c58e...2c58e 5.51.0-1 cooker/main aarch64 bero (bero) mcbin3 2018-10-15 10:41
255361 Build error openmandriva/attica 2c58e...master cooker/main x86_64 bero (bero) predator 2018-10-15 10:43
255374 Tests failed openmandriva/kwayland 5bd03...5bd03 5.51.0-1 cooker/main znver1 bero (bero) c64.openmandriva.org 2018-10-15 10:36
https://abf.openmandriva.org/build_lists/255350

Toolchains

We don’t want to end up in a situation where different architectures have to use different compiler versions, or (even worse) where crosscompilers are out of sync with native compilers…

So we modified our toolchain related packages to generate the native compiler as well as crosscompilers to all other supported architectures from the same source, at the same time.

binutils.spec -- an example

```
%global targets aarch64-linux armv7hnl-linux i686-linux x86_64-linux x32-linux riscv32-linux
riscv64-linux aarch64-linuxmusl armv7hnl-linuxmusl i686-linuxmusl x86_64-linuxmusl x32-linuxmusl
riscv32-linuxmusl riscv64-linuxmusl aarch64-android armv7l-android armv8l-android i686-mingw32
x86_64-mingw32

%global long_targets %(
      for i in %{targets}; do
            CPU=$(echo $i |cut -d- -f1)
            OS=$(echo $i |cut -d- -f2)
            echo -n "$(rpm --target=${CPU}-${OS} -E %%{_target_platform}) "
      done
)

[ … usual rpm boilerplate … ]
```

binutils.spec -- an example

```
%build
for i in %{long_targets}; do
      mkdir -p BUILD-$i
      cd BUILD-$i
      if [ "%{_target_platform}" = "$i" ]; then
            # Native build -- we want shared libs here…
            EXTRA_CONFIG="--enable-shared --with-pic "
      else
            # Cross build -- need to set program_prefix and friends…
            EXTRA_CONFIG="--target=$i --program-prefix=$i- --disable-shared --enable-static
--with-sysroot=%{_prefix}/${i} --with-native-system-header-dir=/include"
      fi
      [...]
      %configure $EXTRA_CONFIG
      %make
done
[...]
```

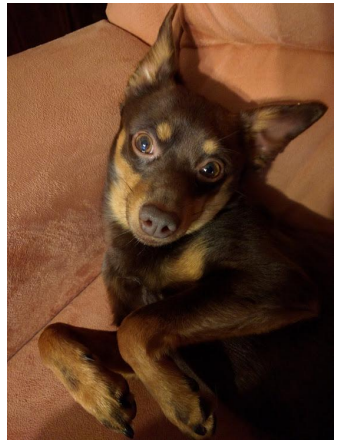
binutils.spec -- an example

```
%(
for i in %{long_targets}; do
      [ "$i" = "%{_target_platform}" ] && continue
      cat <<EOF
%package -n cross-${i}-binutils
Summary:   Binutils for crosscompiling to  ${i}
Group:     Development/Other

%description -n cross-${i}-binutils
Binutils for crosscompiling to ${i}.

%files -n cross-${i}-binutils
%{_prefix}/${i}
%{_bindir}/${i}-*
EOF
done
)
```

Some tweaks for making rpm cross compiler friendly

Wouldn’t it be nice if we could just “rpm -ba --target newarch whatever.spec” on any box (regardless of CPU) to get a whatever.newarch.rpm?

Not 100% there yet, but getting close with a few helper macros.



Detecting that we’re crosscompiling is a good first step:

```
%cross_compiling %(rm -f /tmp/rpm_cc_test 2>/dev/null; echo 'int main() { return 0; }' | %{__cc}
%{optflags} -x c - -o /tmp/rpm_cc_test &>/dev/null; if /tmp/rpm_cc_test 2>/dev/null; then echo -n
0; else echo -n 1; fi; rm -f /tmp/rpm_cc_test 2>/dev/null)
```

Cross-compiling: autoconf

Knowing whether or not we're crosscompiling, we can adjust %configure, %cmake and similar macros to do the right thing

```
%configure \
[...]
%if %{cross_compiling} \
PKG_CONFIG_PATH=/usr/%{_target_platform}/sys-root%{_libdir}/pkgconfig:/usr/%{_target_platfo
rm}/sys-root%{_datadir}/pkgconfig:%{_libdir}/pkgconfig:%{_datadir}/pkgconfig:$PKG_CONFIG_PA
TH; export PKG_CONFIG_PATH; \
CROSSCOMPILE="%{?!noconftarget:--target=%{_target_platform}}
%{?!noconfhost:--host=%{_target_platform}} %{?!noconfbuild:--build=%{_build}}" ; \
%endif \
./configure $CROSSCOMPILE [...]
```

Cross-compiling: cmake

Similar modification for packages that use cmake: Insert

```
CROSSCOMPILE="-DCMAKE_TOOLCHAIN_FILE=\"%_prefix/%_target_platform/share/cmake/%_target_platform.t
oolchain\" -DCMAKE_CROSSCOMPILING:BOOL=ON" ;
```

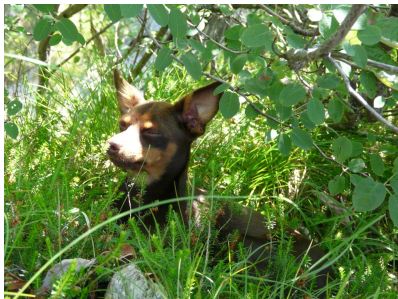
We generate the cmake toolchain file as part of the rpm package:

```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_VERSION 1)
set(CMAKE_SYSTEM_PROCESSOR $ARCH)
set(CMAKE_C_COMPILER "$USR/bin/clang -target $TARGET")
# or set(CMAKE_C_COMPILER "$USR/bin/$TARGET-gcc")
set(CMAKE_CXX_COMPILER "$USR/bin/clang++ -target $TARGET")
# or set(CMAKE_CXX_COMPILER "$USR/bin/$TARGET-g++")
set(CMAKE_FIND_ROOT_PATH "$SYSROOT")
set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM BOTH)
set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

Working around lack of hardware...

Unfortunately, not every package can be crosscompiled cleanly -- some packages build codegenerators and other build tools for the target system and insist on being able to run them.
This can be a problem when boards for new target hardware aren't available yet…

Fortunately qemu 3.x is usually good enough. Qemu built in the -static config, combined with some binfmt_misc setup allows you to essentially chroot into a fake board.

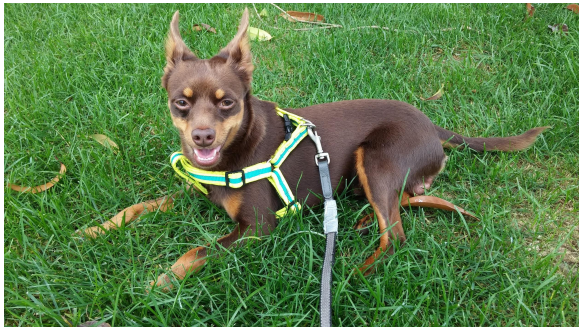


qemu chroots

Qemu chroots can be speeded up by using native binaries for build tools that produce the same result regardless of where they’re run (coreutils, make, …)

When e.g. crosscompiling from aarch64 to riscv64, you can use a riscv64 gcc (producing the right output) combined with aarch64 coreutils, make, etc.

Since clang includes crosscompilers in the same binary without having to change anything, using aarch64 clang is usually safe as well (and much faster than involving the emulator)



Installation

Last big problem: installation -- many non-x86 devices don’t boot from USB or even CD/DVD through UEFI or syslinux…

Unfortunately there is no universal solution yet (but we’re working on it…) - many interesting devices have Android support though, so can we pretend being Android?

Android’s mkbootimg tool (in the AOSP source tree) can generate a fastboot compatible Android-ish boot.img -- even if you feed it with a non-Android kernel and initrd.

Pretending to be Android -- boot.img

Android's mkbootimg tool (in the AOSP source tree) can generate a fastboot compatible Android-ish boot.img -- even if you feed it with a non-Android kernel and initrd.

```
mkbootimg --kernel ${KERNELDIR}/arch/arm64/boot/Image  \
        --ramdisk ${ROOTFS}/boot/initrd-${KERNEL}.img \
        --output boot.img \
        --dt dt.img \
        --pagesize 4096 \
        --base 0x80000000 \
        --cmdline "root=/dev/disk/by-partlabel/system rw rootwait console=ttyMSM0,115200n8
systemd.unit=graphical.target "
```

Pretending to be Android -- system.img

An Android system.img is essentially an ext4 filesystem image containing the root filesystem - but it uses a special compressed format.

It can be generated with `make_ext4fs` (from the AOSP source) - but will lose permissions, ownership etc. to set them up in an Android-ish way…

… unless, of course, it is patched to do the right thing.
https://github.com/OpenMandrivaAssociation/android-tools/blob/master/make_ext4fs-add-keep-uids-option.patch

Pretending to be Android -- system.img

```
make_ext4fs -s -l 10G -K Android/system.img ${ROOTFS}
```

The boot.img and system.img files can now be installed using “fastboot flash boot boot.img” and “fastboot flash system system.img” -- on a device with an Android-ish partition layout that can’t be changed, it’s usually a good idea to mount the userdata partition (can be created with make_ext4fs as well) as /home.

Questions? Comments? Feedback? Dog food for Chwido?

bero@lindev.ch

http://openmandriva.org/

Linaro