Buildroot:DeveloperDaysFOSDEM2014

From eLinux.org

Contents

Buildroot Developers Meeting, 3-4 February 2014, Brussels

What is Buildroot ?

Buildroot (http://buildroot.org) is a set of Makefiles and patches that makes it easy to generate a complete embedded Linux system. Buildroot can generate any or all of a cross-compilation toolchain, a root filesystem, a kernel image and a bootloader image. Buildroot is useful mainly for people working with small or embedded systems, using various CPU architectures (x86, ARM, MIPS, PowerPC, etc.) : it automates the building process of your embedded system and eases the cross-compilation process.

Thanks to our sponsors Google and Mind

We would like to thank our sponsors:

- Google (http://www.google.com) is sponsoring due to their usage of Buildroot to build embedded Linux systems for embedded devices used in the Google Fiber (https://fiber.google.com/about/) project. The source code of their modified Buildroot is available at [1] (https://gfiber.googlesource.com/buildroot/+/master). Google will be providing the meeting location, with Internet connection, but also free lunch and refreshments for the meeting participants.
- Mind (http://www.mind.be) is the Embedded Software division of Essensium, which provides consultancy and services specifically in the field of Linux and Open Source SW for Embedded Systems. Mind is currently hiring! Mind will be offering the Monday dinner to the participants of the meeting.

Thanks a lot to these companies!

Location and date

The Buildroot community is organizing a meeting on Monday February 3rd 2014 and Tuesday Feburary 4th 2014, for Buildroot developers and contributors. This meeting will be a mixture of discussion and hacking session around the Buildroot project. This meeting takes place right after the FOSDEM (http://www.fosdem.org), in order to make it easy for participants to attend both events.

Note that it is not mandatory to attend both days. It is expected that the first day will be mostly dedicated to discussion, while the second day will be mostly dedicated to hacking.

The meeting will take place in Google Brussels offices, located Chaussée d'Etterbeek 180, 1040 Brussels. See http://goo.gl/maps/ZVAGJ for a map of the area.

Participants

From left to right:

1. Jérémy Rosen
2. Thomas De Schampheleire
3. Samuel Martin
4. Ryan Barnett
5. Maxime Hadjinlian
6. Thomas Petazzoni
7. Arnout Vandecappelle
8. Alexander Lukichev (first day only)
9. Peter Korsgaard
10. Yann E. MORIN (photographer).

And in the background, the Google offices.

Report of the meeting

The live edit of the report is at [2] (http://lite2.framapad.org/p/N41iEOV4Gi).

Summary

- Thomas P. will post the patches for the BR2_PREFER_STATIC_LIB refactoring. They may be merged in -next, but the one that adds the prefer-dynamic option should get more (runtime!) testing before acceptance.
- A full chain of trust setup is too complex for us. However, we will add support for hashes of downloads - preferably the one published by upstream.
- Maxime H. will cook some patches so that SXX* files in the package directory are installed to /etc/init.d, and *.service files are installed to multi-user.wants.
- Thomas DS. will extend the documentation with an explanation of the meaning of Acked-by, Tested-by and Reviewed-by. Explanation of Tested-by goes to the beginning of the Contributing section, to encourage people to do testing.
- Arnout will create a crontab script that gathers statistics from patchwork, so we can evaluate the evolution.
- One of the Thomases will regularly send out mails which summarises for two or three of the "difficult" patch sets what action needs to be taken on them.
- When we see patches that will probably take a long time before acceptance because they touch core infrastructure, we will warn the contributor of this.
- Thomas P. will take over commit access when Peter is away for a short time (e.g. a weekend).
- Thomas P. will clean up and publish his autobuilder scripts, in the hope of attracting more autobuilder hardware. Other autobuilder features like testing additional branches is delayed until we have more horsepower to work with.
- Yann will evaluate the Pengutronix genimage tool and see if it can be used as a basis for the genimages infrastructure. If not, the genimages script will

be distributed as a separate package.
- We continue bugging the uClibc maintainers to make a release.
- When there are autobuilder errors due to uClibc being out of date, we either add a package-specific patch if that is easy, or we mark the package !UCLIBC.
- Thomas P. will send a mail to the list that proposes the two GSoC topics we already have, and asks for other suggestions.
- Jérémy may post a patch that points .config to some other place (under version control).
- Someone **(who?)** adds to the documentation how defconfigs can be merged, and that this should be done in a wrapper script or BR2_EXTERNAL Makefile.
- Thomas DS. cleans up the python depends vs select, adding explicit config options when a non-python package has an optional python dependency.

BR2_PREFER_STATIC_LIB refactoring

Thomas P. has the idea to create three options: dynamic-only, static-only, and prefer-dynamic (= what is currently done). He has a set of patches for that (not yet submitted to the list). The first two are bugfixes so can be applied. With dynamic-only, Thomas had some builds which were sped up by 25%. The main thing that remains to be done is testing, because the dynamic-only may cause some issues. E.g. ThomasDS found out that tcpdump links statically against libpcap.

Another issue is the usefulness. If you want static libraries, you typically don't want it for everything, but just for some libraries which are more efficient to link statically. So you would need a per-package option to enable static/dynamic. Actually there are other options that we would like per package, e.g. debug symbols, stripping. Buildroot doesn't have a solution for this. local.mk is an option, but it's not very good because it is not considered part of the configuration.

Idea of Jérémy: in the infrastructure, add targets **libfoo-static** and **libfoo-dynamic** so that a package can make sure it gets the library it needs.

Conclusion: first two patches to be merged for -next, the last one to be merged later. The last one first needs an allpackageyes test. Default should stay at current situation, down the line we can change default to dynamic-only.

How to create a chain of trust for the verification hashes

First idea: add a hash of the downloaded package to buildroot, and verify it after download. However, there should also be a chain of trust between the hash and the final signature by Peter when he makes a release. We had some discussion with different ideas of how things can fail, but the general feeling was that we are adding complexity and we're still not bullet-proof.

- Patches are sent with a package hash.
- Reviewers check that the hash is one that was published by upstream.
- Peter commits, makes a release and signs it.
- Buildroot downloads the package and verifies the hash locally.

It's not perfect, there are still a few attack vectors possible, but it works for us and it's an improvement over the current situation. If a user is really picky about security, he will still have to check himself but the hashes give him an easier way to gain trust. The latter should be clarified in the manual.

Conclusion: add optional hash checking to package infrastructure, document how it should be used, but nothing more. Yann is taking care of pushing this topic forward.

SystemV/systemd init scripts

The idea is that the infrastructure checks the package directory for the existence of SXX resp. XXX.service and installs it in the usual place. Problem with systemd unit files is that they may have to be installed in a different place than the default multi-user.wants. However, most attendants to the meeting felt that such customization is typically done in a post-build script, so we shouldn't worry about it.

If the _INSTALL_INIT_XXX hook exists, both are done.

Conclusion: Maxime H. will cook up patches for this.

Meaning of Acked-by/Reviewed-by

Thomas DS posted a proposal on the list:

- Tested-by: as in the manual: I performed some kind of test (typically described below the tag) on the patch.
- Reviewed-by: I code-reviewed the patch and did my best in spotting problems, but I am not sufficiently familiar with the area touched to provide an Acked-by. This means that, although I reviewed the patch, there may be remaining problems that would be spotted by someone with more experience in that area. The detection of such problems should not mean that my Reviewed-by: was too hasty.
- Acked-by: I code-reviewed the patch (note: not necessarily tested) and am familiar enough with the area touched that I can indicate it is a good patch. If someone else detects a serious problem with this patch afterwards, then this Acked-by may have been too hasty.

Amendments:

- If you have some comments and the patch is good to go after these changes, **don't** put an Acked-by. Acked-by means it's good to go **as it is**.
- Anybody can review, only a newcomer's opinion will not be trusted as much. This should be better clarified in the manual.
- In addition to the formal tags, people should reply to infrastructural changes whether they agree to the principle with a simple English sentence.

Conclusion: ThomasDS will add these clarifications to the documentation.

Side comment: patches should be split so that infrastructural changes are separate from the package changes. Except of course if the package changes have to be done together with the infrastructure. Some people think that documentation should be in a separate patch, but others disagree - so it's on a patch-by-patch basis. The majority prefers documentation changes separately, though.

Ideally, patches should be simultaneously bisectable and reviewable (i.e. split up). Sometimes this is impossible (e.g. wayland/weston version bump). In that case, reviewability should win (e.g. wayland and weston bumps as separate patches). Peter may squash a series into a single patch if he feels like it.

Evaluation of the patch acceptance process

Ideally we should have statistics about incoming and outgoing patches. Patchwork doesn't really have a record of this. Idea: cron job that counts how many patches are in each state every day, and keep this in a separate DB. Statistics should also track how old patches are, but that's fairly easy to do in patchwork. What we really want to know is the delta between new and accepted/rejected/superseded patches.

Follow-up of old patches: ThomasDS would like the patchwork queue to be cleaned up sufficiently so that at the time of a release, there is nothing remaining from before the previous release, so there is never anything more than 3 months old. The main goal is to create a sense of urgency about these patches, similar like we have for the autobuild failures during the release month.

For the patchwork cleanup, it's important not to reject patches blindly. There are always a few that would be good to keep around, even if nobody currently has time to review or test them. ThomasDS currently delegates these things to himself, with the idea that he will do this later. We can add a link to the todolist on elinux that filters the delegated-to-Thomas patches from patchwork. A link has been added along with a short write-up at [[3] (http://elinux.org/Buildroot#Todo_list)]

For the "difficult" patchsets that keep on hanging around, ThomasDS proposes to also regularly send out mails that summarize the state of these patchsets: what still needs to be changed, what still needs to be tested. The main point is to bring them back into the attention field. The most important thing is that we avoid that the author has done his homework, but that the buildroot community just doesn't react anymore. This should bring the focus to just one to maximum three patchsets.

For these difficult patches, it helps a lot if a defconfig + README is provided that allows testing in qemu.

Conclusion: Arnout creates a script that tracks the number of patches in patchwork every day, ThomasP puts this in a cron job on the autobuild server. Should be something like (for 'New' state): pwclient list -s New | wc -l **For the "difficult" patchsets, we delay until next meeting (plenty of time before we get there). In the mean time, though, we'll try to send around 'status' mails for the big patchsets, indicating why they are not yet accepted (for example they need more testing, they need a decision from the community, we are waiting on the author, ...). These mails should make sure focus is kept on these patches. When a patch is pinged, it should be a big alert for us that we should give some kind of reaction. When reviewing patches that are expected to take a long time to get in, state in the review that it will take some time so the original contributor knows what to expect. ThomasP will take over commit access also when Peter is just away for a short time (week-end).**

To get more people to post reviews, the solution is to credit them better - currently reviews are not really credited. Acked-by can be grepped in the logs, but reviews which don't lead to an Acked-by are a bit more difficult. Still, giving credit for Acks is useful. And more so for Tested-by. Perhaps in the manual's Contributing section we should put adding of Tested-by at the top. Also when a newcomer does a Tested-by (or Acked-by), we should send them a thank-you mail. We should update the documentation so that it shows the steps for contributing in the right order: 1. do tests of patches; 2. look at the todo list; 3. send patches.

Peter will add these 'rankings' to the release e-mails.

Autobuilder wishlist

Thomas will cleanup and post his scripts so other people can use it.

What do other people would like to see in the autobuilders?

- Run-time tests - this is a big topic
- Expand the number of autobuild machines - depends on the scripts that Thomas will post. Would be nice to ask the companies that use buildroot to install an autobuilder. Perhaps make a VM image that they can use, so there is no risk of leaking sensitive information.
- Click on a package and see the last failures of that package. Ideally also the successes, but that's difficult.
- Expose (readonly) SQL queries to the database.
- Thomas should post the layout of the database, then we can submit patches to the PHP crap with new SQL queries for other views.
- Jenkins is running on the defconfigs, this seems to be enough. If you want to be spammed by it, ask Maxime Ripard.
- Maybe we can attract hardware donations to add autobuilders? (Publish requirements, space, cpu, bandwith, ...)
- Between releases, it would be possible to use the -next branch to test risky items.
- Autobuilding of the next branch: do we even want this? Shouldn't we focus on fixing the release?
- If we test multiple branches, we probably want a separate instance of autobuild.buildroot.net - that solves all technical problems.

Conclusion: we first need more autobuild infrastructure, because with less than 100 builds per day the build results are not very relevant. The other items will be started by ThomasP in the hacking session.

Genimages

Remark from Peter: buildroot doesn't normally create new tools, just infrastructure for building stuff. It would be a pity if something that is useable beyond buildroot would be hidden in the buildroot infrastructure and not used by anybody else. The genimages tool looks like it could be something independent from buildroot, and actually something like that exists in Pengutronix as well. OTOH, we certainly don't want an ad-hoc solution for every different platform (like we now have something for iso images and something for RPi). Yann thinks that for something like genimages, C is not an approprate language, it should be a shell script instead. Still, Yann will investigate if the Pengutronix genimage tool is useable. They are also generally very cooperative so if we need to make changes they are likely to accept them.

ThomasP thinks that the complexity of the tool that Yann added was too high, but OTOH this tool is supposed to reply the "do your own thing" that we tell people to do in the postimage script, so it really should cover all use cases.

Conclusion: Yann will evaluate genimage and if necessary contact the maintainers. If a new genimages is still needed, it should be a separate package.

How to handle the uClibc problem

The main issue is that uClibc is almost dead - they don't do releases anymore (0.9.33.0 is two years old, and there are more than 50 patches on the 0.9.33 branch since the 0.9.33.2 bugfix release) . There are many features that are

needed to build modern packages, for which we carry patches. The end result is that there are packages that build with the internal toolchain, but not with an external toolchain (e.g. Blackfin toolchain, old buildroot toolchain, crosstool-NG toolchain). We have had package-specific patches but also patches to the internal uClibc toolchain, and these may conflict with each other. In addition, ThomasP has asked multiple times on the uClibc list to make a release but they're not doing it.

Even if there is a new uClibc release, the problem is not completely solved because we have no way to know which uClibc is used by the external toolchain. It's a bit similar to the uClibc config: we don't know what the external toolchain contains. But in practice, that hasn't caused many problems; most of the problems have been caused by the difference in patches.

ThomasDS launched the idea of creating a supplementary library that adds the missing things in uClibc. Unfortunately this can't solve everything, because sometimes there are changes in headers etc.

Switching to glibc as the default has the problem that it makes the build very long. For the first-time experience, this is problematic because the build speed is one of our USPs. Plus, some architectures (Blackfin, Xtensa) will only ever be supported by uClibc. A second evaluation seems to indicate that the build time difference is not so large at all...

We could ask Analog to build a new toolchain with the necessary uClibc patches, but communication with Analog hasn't been that great.

We can send a message to the uClibc mailing list saying that if there is no release forthcoming, we consider dropping the uClibc support.

Since external toolchains (including crosstool-NG) will rely on vanilla uClibc, we really should support them properlly and it's not sufficient to just warn users that with an external toolchain it's not guaranteed. And if we mark all packages that new features as !UCLIBC, then we end up with the situation that our suggested default (internal uClibc toolchain) doesn't actually build half of the packages (because they've been marked !UCLIBC).

It's still possible to make a fork of uClibc that does nothing more than make releases.

We can add an extra config symbol that marks the uClibc as a buildroot-built one or an externally-built one. For an internal toolchain, it is selected automatically, for an external toolchain the user has to mark it. The help text should explain what the implications are of marking this option. Still, it seems to be too much of a burden compared to the number of problems that really occur.

Conclusion: First of all, we continue bugging the uClibc maintainer. Second, we keep the feature patches to the internal uClibc unless they cause a problem. Third, when there are autobuilder errors, we either add a package-specific patch if that is easy, or we mark the package !UCLIBC.

Website and branding

Should we have a booth at FOSDEM? The problem is that there have to be people that man it, and the amount of exposure you get from it is probably limited. Perhaps we can put some posters on the walls, to improve the brand awareness.

There are quite a few buildroot videos on youtube or in other places. We should link to these from the website; same for the slides etc.. It would also be nice to make a screencast of a buildroot training, but that's really a lot of work.

ThomasP has the seed of a new website that needs some brushing off. If he publishes it, other people can work on it as well. Perhaps something to hack on tomorrow.

Google summer of code

Submission of the organisation starts today. ThomasP will start this process.

Topics to propose:

- Continue the work done by Spenser: integration of OpenGL stuff, because we (= Spenser) anyway still has the board.
- Improvement of test infrastructure with specific test cases (e.g. kernel) and runtime tests (in qemu). This could potentially include target rootfs verification (see TODO list).

We looked at the other items on the todo list, but most of it is either too small for a GSoC project, or too difficult, or not really sexy. Also, we prefer to retain topics that we really want and not add a third one that has lower priority. However, it's possible to group a number of smaller, partially related tasks under one umbrella GSoC.

ThomasP has identified a number of forks and it would be nice to grab the most useful additions from these.

Criteria for the next student:

- Full-time availability in July and August
- Preferably in Europe
- Availability in evenings (2 per week?) to have some live interaction with the mentor.

ThomasP will send a mail to the list mentioning the two selected topics and asking for others.

Evaluation of BR2_EXTERNAL

Jeremy explains how he uses buildroot. He makes a final product (not a bare board), wants to store anything in git, and has stupid users. How he works is that he has a git repo with buildroot as a git submodule. All the things which are particular to his project are kept in the top directory (extra packages, fs overlay, defconfigs, support scripts). The buildroot directory itself should ideally be read-only (soon it will be possible). With BR2_EXTERNAL, this scheme works. The only tricky thing is saving the configuration: you have to explicitly do a savedefconfig and linux-savedefconfig. In addition, if somebody does a savedefconfig and I don't think to update my config, I don't get any warning from git when I'm doing a savedefconfig myself. Basically, Jeremy would like to have the .config be under version control. The same is true for the other configs.

You can consider menuconfig as an editor of a file (.config or defconfig) so it makes sense that the result of this editing is saved in git. What we could do is to automatically do a savedefconfig whenever a config is edited. The savedefconfig should then also be patched to save back to BR2_DEFCONFIG. But then, when a user does 'make rpi_defconfig' and then starts changing stuff, he doesn't want to see changes in his buildroot git.

The behaviour of buildroot config is something we inherit from the kernel etc. So if we deviate from that, we will surprise our users.

There's nothing stopping you to create a Makefile or script in your toplevel directory that does the automatic saving.

What we certainly can use is a single command that saves all config files (kernel, busybox, uClibc, barebox). To do this well, we have to be able to detect if a config is the default one or not; for that, we can add an extra option to busybox so the user has to explicitly say if it is the default or not. But see also "Handling of make xxx-menuconfig" in [[4] (http://elinux.org/Buildroot:DeveloperDaysFOSDEM2013)]

The biggest problem is educating the users, and mistakes will happen.

What we would certainly be OK with is that the path to a config file in some other place than $(CONFIG_DIR) can be given on the command line. Jeremy has to think a little about what to do with the linux and other configs.

Conclusion: Jeremy may post a patch for pointing .config to some other place, otherwise no action.

State of major patch sets

Systemd/udev support (Eric Le Bihan)

Problems with this patch set:

- Name change of eudev selection symbol -> can be handled by legacy
- Needs final review (v6 reviewed on Monday by ThomasDS, feedback sent to Eric)
- Has impact on several packages that depend on udev, so needs careful testing
- 2-3 people work on this series tonight

It's a candidate to still be merged in 2014.02, because our current udev/systemd support is really bad now.

Perl package infrastructure (François Perrad)

Reviewed by Yann and Arnout.

SELinux support (Clayton Shotwell)

Not looked at.

libdrm/mesa3d updates (Bernd Kuhls/Spenser Gilliland)

Thomas P. will review this.

How to handle 'target' defconfigs versus 'development' defconfigs

The basic ide is to be able to split a defconfig in the generic part and a feature part. There are many different use cases of this. In any case, you have two problems:

1. Two (or more) defconfig fragments have to be combined. This could just a cat of the different fragments. But ThomasP has a customer that would combine fragments in a more intelligent way. There is in fact a script in buildroot that allows to combine defconfig fragments and supports duplicate entries (support/kconfig/merge_config.sh).
2. Save the defconfig. This is a fundamentally unsolvable problem because a human being will have to put the changed part in the right fragment. You could add some pseudo-intelligence that finds the fragment which manipulates this option and then updates it; for options that don't appear in any fragment, they would be added to the "top" one. The kernel has a script (scripts/diffconfig) that diffs two config files intelligently, which can help for making the split.

Conclusion: add to the documentation an explanation of these two issues and that it can be added to a BR2_EXTERNAL .mk file or to a wrapper script.
Side remark: the kernel has a script (scripts/config) that updates config options, which is probably better than our own KCONFIG_SET_OPT and friends.

Config.in.legacy for packages that grow a dependency

The existing legacy handling works for the removal of a config option. However, if you are currently using a package, then upgrade to a newer buildroot version in which this package depends on a toolchain option (or gets another dependency) that you do not have enabled, the previously-enabled package is suddenly invisible. It would be nice if this problem can be detected and mentioned to the user, but there do not seem to be a clean solution to detect this.

The user should generate a new defconfig after the update, and compare this to his original defconfig. Any difference here should be reviewed.

Yann makes the suggestion to detect which options enabled in the defconfig are _not_ enabled in .config (after parsing by kconfig) and warn the user about such discrepancies. Though this is feasible, it will not detect every problem and it seems a bit complex to implement. So no action for now.

Conclusion: no action.

Python packages: depends vs select

ThomasP hosted a patch series that creates three categories of python packages:

- Python modules depend on python - this is the case already
- Programs that use python should select python - a few have to be changed
- Libraries that have python bindings: sometimes they can be generated automatically if PYTHON is selected, sometimes there is a config option, sometimes bindings are never generated. In general, bindings are significantly smaller than the library and python itself. There's something to be set for all solutions, but in the end it was decided that the simplest is to always have an explicit option for the python dependency.

On a related note, some packages (e.g. python-protobuf) use exactly the same source as another package. Currently this means copying the _SITE etc. assignments, or relying on the order of evaluation. Peter suggests that in such cases, we can put the two packages in the same .mk file so the order is guaranteed. Arnout would like to see infrastructure to support this use case explicitly, so the infrastructure will delay evaluation appropriately so it doesn't depend on order of evaluation.

Conclusion: we change the handling of optional python bindings: instead of selecting them automatically when python is enabled, we add explicit config symbols for them. ThomasDS will update the patches accordingly.

pkgparentdir removal

[[5] (http://patchwork.ozlabs.org/patch/290254/)] removes the pkgparentdir helper function which is not really needed. This can just go in. However, this patch also changes the meaning of <PKG>_DIR_PREFIX. See the link for the details. The idea is to clean up the different variable names. What is important is to limit the number of variables that are defined, so we decided to try to avoid creating a variable for $(TOPDIR)/package/foo; instead always explicitly use package/foo/xxx, and $(pkgdir) in the infrastructure. <PKG>_DIR_PREFIX can be removed (possibly in a separate patch)

Related to this change are the variables FOO_SRCDIR and FOO_BUILDDIR. It looks like these are redundant: the few places that use these variables can be replaced by their definition. This avoids slowing make down by adding more variables. This change, however, is not urgent. We also discussed whether FOO_SRCDIR should point to the root directory of the package source tree, or whether it should include FOO_SUBDIR (as today). The general agreement seems to be that FOO_SRCDIR should be changed to point to the root directory of the package source tree, and all the package infrastructures that need to use a subdir should do $(FOO_SRCDIR)/$(FOO_SUBDIR)/blah.

Write down the rules for the prefixes

Refer to: [[6] (http://lists.busybox.net/pipermail/buildroot/2014-January/087995.html)] Currently we use the following:

- BR_ prefix seems being used mostly for internal toolchain (wrapper) stuff;
- BR2_ prefix: widely used in all Config.in;
- BUILDROOT_ prefix: for things that can be set from the environment;
- PKG_ prefix: in my mind it's to close to what we can set for pkg-config - I don't really like this one.

The proposed rules are:

- BR2_ for Kconfig options. There will be some environment variables that have the same name as a Kconfig option (e.g. BR2_EXTERNAL). BR2_ should not be used for normal make variables. **ThomasDS will check if this is done consistently.**
- BR_ prefix for things that are used internally (not user-visible). **BUILDROOT_LIBC, BUILDROOT_COMPILER_CFLAGS, BUILDROOT_COMPILER_CXXFLAGS have to be replaced with BR_xxx**.
- BUILDROOT_ prefix for things that are settable through the environment but not in the config. There are two problematic cases: BR2_EXTERNAL (which is normally set through the environment, not user-visible in .config) and BUILDROOT_DL_DIR (which is an environment variable that overrides BR2_DL_DIR).

Conclusion: don't use BUILDROOT_ anywhere, but use BR2_ for Kconfig variables, BR2_ for user-facing environment variables and BR_ for internal environment variables. Add something to Makefile.legacy to warn the user of old environment variables. Arnout will make these changes.

title=Buildroot:DeveloperDaysFOSDEM2014&oldid=347504"
