Patching Unpatchable Files

Anna-Lena Marx

Yocto Project Summit

December 5th, 2024 · virtual

inovex

Anna-Lena Marx

Anna-Lena Marx

anna-lena.marx
@inovex.de

marx.engineer

Embedded Systems Developer

- since 2015 with inovex
- has a Master's degree in Embedded Systems
- studies Electrical Engineering as a hobby

Main Topics

- Embedded Systems
- Yocto Linux
- Linux Kernel
- AOSP/AAOS
- IoT

inovex

What do I mean with “unpatchable” files?

inovex

.bbclass files (Classes)

- Used to abstract common functionality
- Shared between multiple recipes
- Used via **inherit** in a recipe

Well-known classes

- archiver
- autotools
- cmake
- cve-check
- image
- kernel
- ...

inovex

.inc files

- Shared between recipes to encapsulate common behavior
- Always part of a recipe via **include** or **require**
  ... causes BitBake to parse whatever file you specify,
  ***and to insert that file at that location**.*

If used in a .bb file

-> just a part of a recipe

-> can be overwritten using .bbappend for a specific recipe!

Special case: .inc files used in bbclass files
-> part of this talk

inovex

Hold on a second...

Do you really need to patch a bbclass file?

Valid cases for patching .bbclass files

- When copying to your own layer does not make sense,
  e.g. the class is used in the original layer itself
- If you want to keep track of your (LTS) upstream and stay as close to the original as possible
  e.g. while working on submitting the change

Why did I want to modify an existing .bbclass file in first place?

```
diff --git a/meta-mender-core/classes/mender-artifactimg.bbclass
b/meta-mender-core/classes/mender-artifactimg.bbclass
index cdb07551..3dd2a4ba 100644
--- a/meta-mender-core/classes/mender-artifactimg.bbclass
+++ b/meta-mender-core/classes/mender-artifactimg.bbclass
@@ -130,7 +130,7 @@ IMAGE_CMD:mender () {
     mender-artifact write rootfs-image \
         -n ${MENDER_ARTIFACT_NAME} \
         $extra_args \
-        $image_flag ${IMGDEPLOYDIR}/${ARTIFACTIMG_NAME}.${ARTIFACTIMG_FSTYPE} \
+        $image_flag ${IMGDEPLOYDIR}/${ARTIFACTIMG_NAME}.${ARTIFACTIMG_FSTYPE}${DM_VERITY} \
         ${MENDER_ARTIFACT_EXTRA_ARGS} \
         -o ${IMGDEPLOYDIR}/${IMAGE_NAME}${IMAGE_NAME_SUFFIX}.mender
 }

...
```


inovex

The official solution for *bbclass files*

inovex

“”

*This is simply because **you can't patch a bbclass**.*
*You'll have to **copy it into your layer with the modification**,*
*or **work upstream** to get a suitable change merged.*

Ross

https://docs.yoctoproject.org/pipermail/yocto/2018-October/042858.html

inovex

Upstream your work

Always the best solution.
Your work could be relevant for others, too!

But upstreaming and maybe backporting work to all LTS versions takes some time.

inovex

Copy the .bbclass file into your own layer with modifications

Generally the recommended solution in Yocto!

- Full control over the whole class
- Fully responsible in keep it the copies in sync with upstream layer

Caution!

During a build, the OpenEmbedded build system looks in the layers from the top of the list down to the bottom in that order.

For proper overwriting, your own layer needs to be above the one to overwrite!
Layer priorities alone do not change that!

inovex

That would have meant to me

Enable Mender using system partitions with dm-verity

- *mender-artifactimg.bbclass* controls image creation
  -> needed modification

- *mender-part-images.bbclass* creates WKS file during compile time
  -> needed modification

- Both used in *mender-setup.bbclass*

So why did I not like that approach for my problem?

A simple change would lead to copying **three .bbclass files** into an own layer (in this special case)

-> Vendoring

- History and branch/tag information is lost
- Divergent states between vendored version and upstream
- No upstream updates without further (manual) actions
- Maybe strange errors if not noticing the original file has changed

Is there anything better?

Variation of the full file copy

- Create a .bbclass file with the same name in your layer
- Inherit from the original
- Override the target function

But still same vendoring issues as before but with a reduced scope.

Does not work in each case.

Not every idea is a good one

> *Regarding patches:*
> ***any files with .patch extension that you add to the SRC_URI variable will be applied after fetching and unpacking all sources.***
> *It doesn't matter how/where you add these files, it can be in the original .bb, .bbappend, .inc or even in .bbclass files.*
> *The main point is the SRC_URI variable.*

*I've **created a patch file in our bsp layer of the same name as the bbclass** file I want to patch but the patch is not being applied.*

https://stackoverflow.com/questions/77018036/need-clarification-regarding-when-to-use-bb-inc-file-in-yocto
https://docs.yoctoproject.org/pipermail/yocto/2018-October/042857.html

inovex

meta-mylayer/some-software
source directory

do_fetch

do_unpack

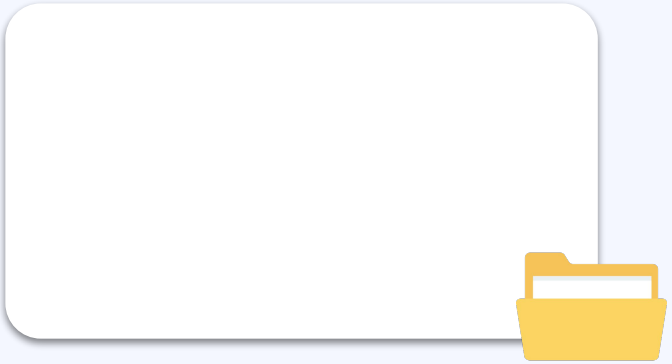
do_patch

do_configure

do_build

do_install

do_package



build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex

meta-mylayer/some-software
source directory

do_fetch

do_unpack

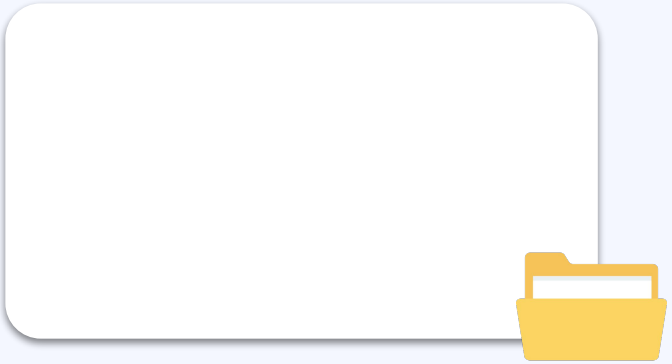
do_patch

do_configure

do_build

do_install

do_package

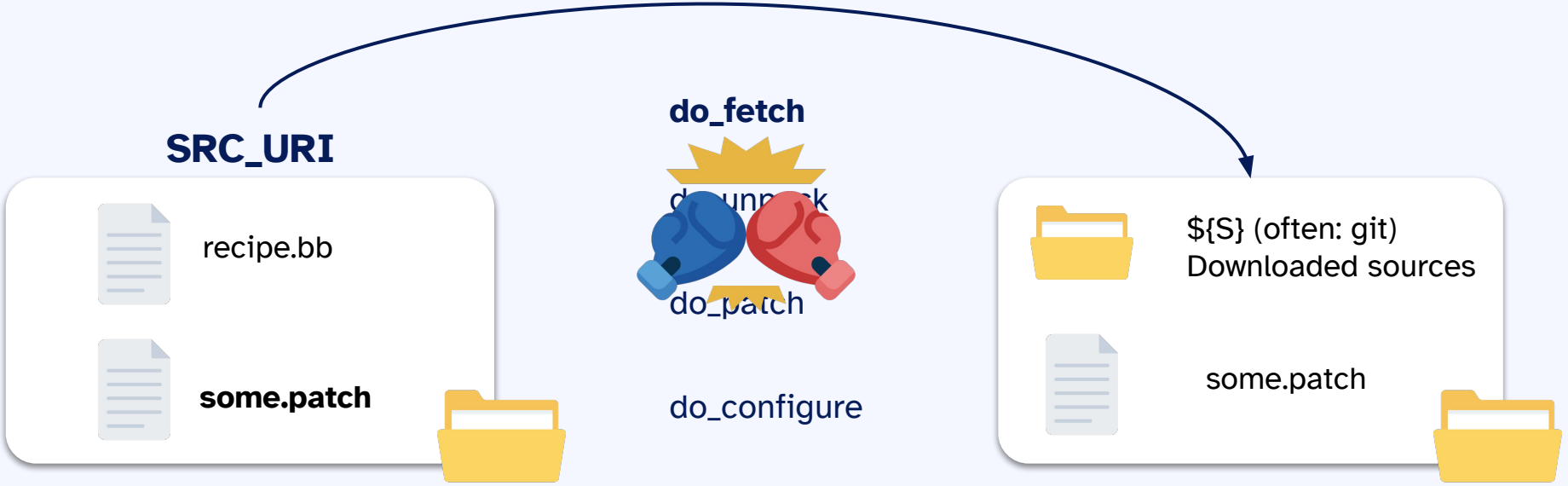


build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

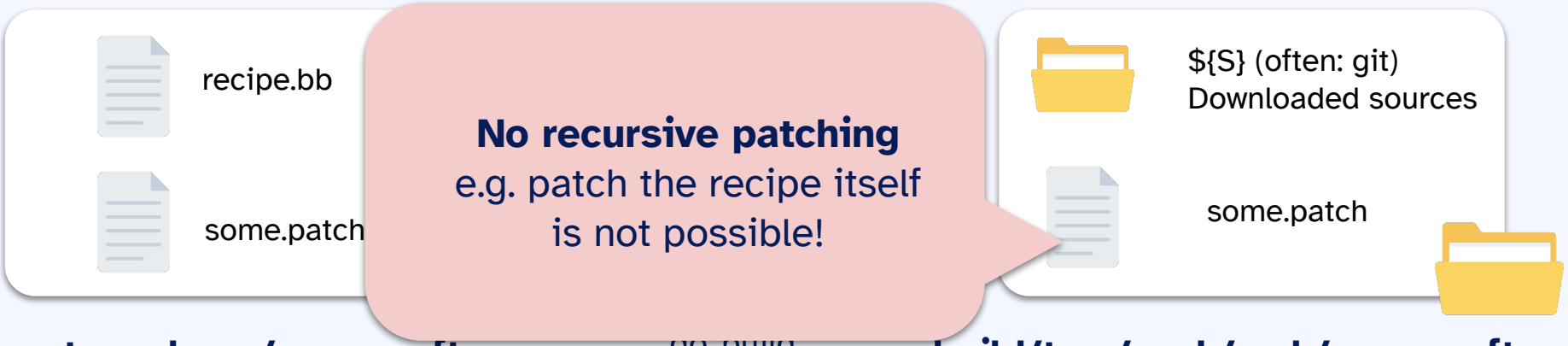
inovex


SRC_URI
recipe.bb
some.patch
meta-mylayer/some-software
source directory
do_fetch
do_unpack
do_patch
do_configure
do_build
do_install
do_package
${S} (often: git)
Downloaded sources
some.patch
build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex


SRC_URI
recipe.bb
some.patch
meta-mylayer/some-software
source directory
do_fetch
do_unpack
do_patch
do_configure
do_build
do_install
do_package
${S} (often: git)
Downloaded sources
some.patch
build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex



meta-mylayer/some-software
source directory

do_fetch

do_build

do_install

do_package

build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex

meta-mylayer/some-software
source directory

do_fetch

do_unpack

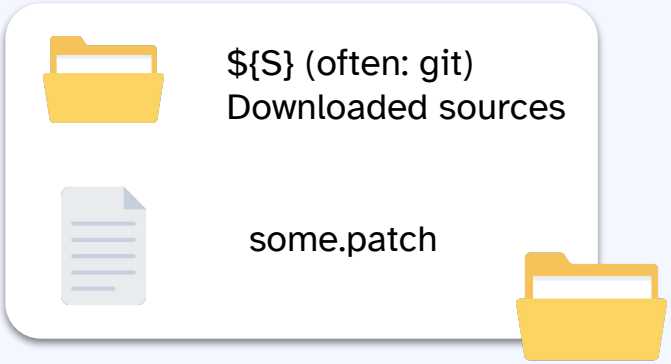
do_patch

do_configure

do_build

do_install

do_package



build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex

do_fetch

do_unpack

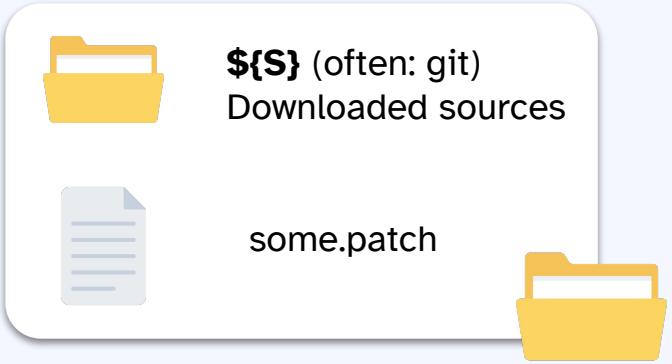
do_patch

do_configure

do_build

do_install

do_package



build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex

do_fetch

do_unpack

do_patch

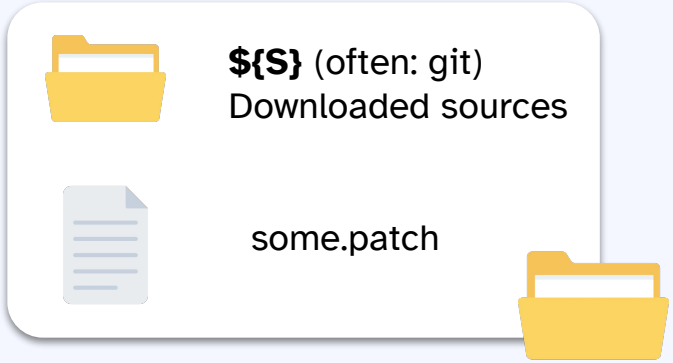
do_configure

do_build

do_install

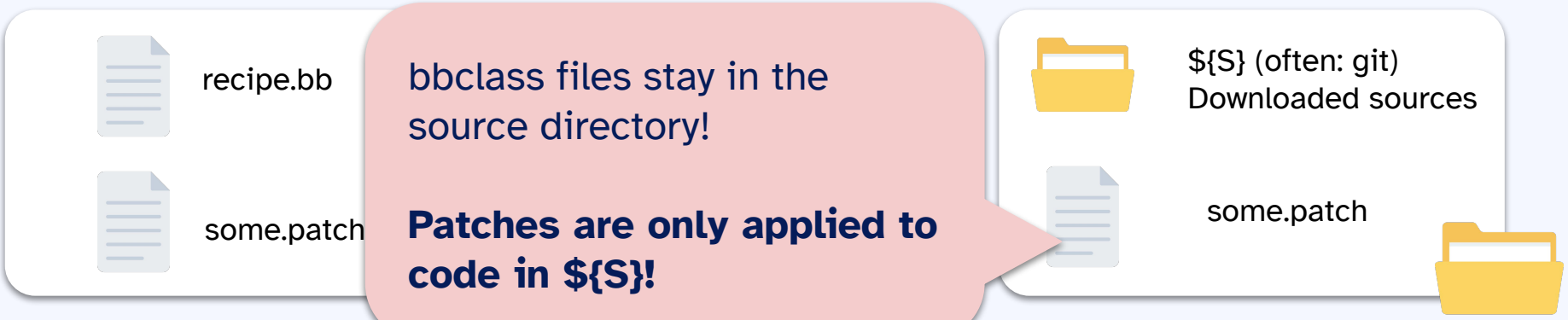
do_package

do_patch applies patches to the recipes source files which are located in **${S}**!



build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

inovex



meta-mylayer/some-software
source directory

do_fetch

do_build

do_install

do_package

build/tmp/work/arch/some-software
${WORKDIR}/${BP}-${PV}
${B}

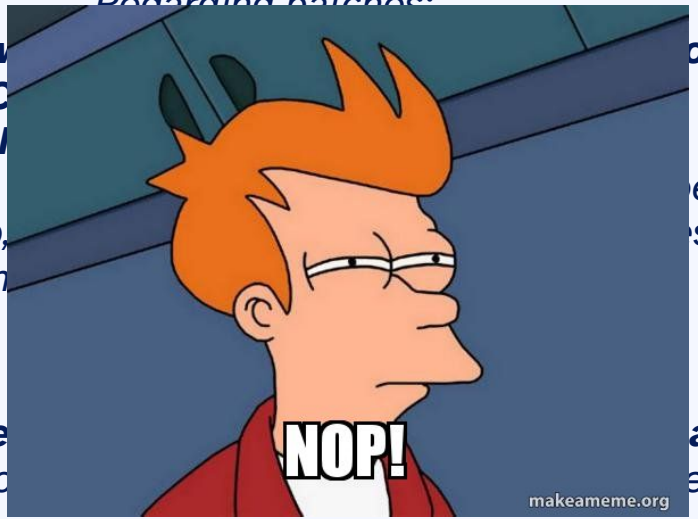
inovex

Not every idea is a good one

Regarding patches:

any files w ***o***
the SRC
fetc
It doesn't matter *e in the*
original .bb, *s.*
The n

I've ***created a patch file*** ***as the bbclass***
file I want to *ed.*



https://stackoverflow.com/questions/77018036/need-clarification-regarding-when-to-use-bb-inc-file-in-yocto
https://docs.yoctoproject.org/pipermail/yocto/2018-October/042857.html

inovex

KAS Patching Mechanism

- Available in KAS, but barely documented
- Intended use case unknown

- Exactly the thing I was looking for
  - Really clean
  - Maintainable
  - Integrates in the existing workflow
  - Applying the patch will fail on major upstream changes
    -> but this is what I wanted

- And I used KAS already

https://kas.readthedocs.io/en/3.0.2/userguide.html?highlight=patch

kas/common.yml

```
    meta-mender:
        layers:
            meta-mender-core:
        branch: kirkstone
        url: https://github.com/mendersoftware/meta-mender.git
        path: sources/meta-mender
        patches:
          dm-verity:
            repo: meta-vgrid
            path: patches/meta-mender/mender-include-dm-verity-image.patch
```

meta-vgrid/patches/meta-mender/mender-include-dm-verity-image.patch

```
diff --git a/meta-mender-core/classes/mender-artifactimg.bbclass
b/meta-mender-core/classes/mender-artifactimg.bbclass
index cdb07551..3dd2a4ba 100644
--- a/meta-mender-core/classes/mender-artifactimg.bbclass
+++ b/meta-mender-core/classes/mender-artifactimg.bbclass
@@ -130,7 +130,7 @@ IMAGE_CMD:mender () {
     mender-artifact write rootfs-image \
         -n ${MENDER_ARTIFACT_NAME} \
         $extra_args \
-        $image_flag ${IMGDEPLOYDIR}/${ARTIFACTIMG_NAME}.${ARTIFACTIMG_FSTYPE} \
+        $image_flag ${IMGDEPLOYDIR}/${ARTIFACTIMG_NAME}.${ARTIFACTIMG_FSTYPE}${DM_VERITY} \
         ${MENDER_ARTIFACT_EXTRA_ARGS} \
         -o ${IMGDEPLOYDIR}/${IMAGE_NAME}${IMAGE_NAME_SUFFIX}.mender
 }
...
```


inovex

Other management tools

Repo / bitbake-layers (create-layers-setup) / ...

- No support for applying patches

Own script or hook

- Mixed feelings
- Another tool
- Maybe error-prone

inovex

To sum it up

inovex

What strategy is the best for you?

Don’t patch .bbclass files

Upstream your changes

Copy the bbclass file to your own layer /

Use KAS or additional tooling/scripts

inovex

Remember

- Don’t use patching as a general solution to work with Yocto at all!
- Don’t use patching over existing, well-known workflows like .bbappend!
- Keep in mind who will maintain a project in the future, minimize pitfalls

If overwriting a bbclass file by copying to your own layer does not work out, check the layer ordering, not only their priority!

In the end

- A solution must fit your (and the project's) needs
- Don't introduce a tool like KAS if not using it already
- But don't create your own workflow and tooling from scratch, there is already a large zoo of management solutions and tools

I would love to see something like .bbappend for .bbclass files directly inside Yocto/OpenEmbedded in future!

But - how to avoid misuse?

Do you utilize or know about other solutions or strategies?

inovex

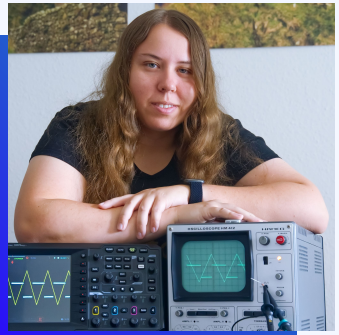
Thank you!

Time for questions and discussions!

inovex is an IT project center driven by innovation and quality, focusing its services on ‘Digital Transformation’.

- founded in 1999
- 500+ employees
- 8 offices across Germany

www.inovex.de



Anna-Lena Marx
Embedded Systems Developer

anna-lena.marx@inovex.de

inovex