Demystifying the OVERRIDES mechanism and Bitbake operators 2022 edition

Quentin Schulz
Theobroma Systems Design und Consulting

Yocto Project Summit, May 2022

About

About

- BSP engineer @ Theobroma Systems Design und Consulting
- Open positions: https://careers.theobroma-systems.com/job/
- Design and produce smart embedded systems for IoT
- Concept to mass production and long-term support
- Expertise in custom HW development, embedded security, performance engineering and creation & support of BSP for Linux
- quentin.schulz [at] theobroma-systems.com
- foss+yocto [at] 0leil.net (zero not o)

Why a re-edition?

- New OVERRIDES syntax! (: instead of _)
  - New syntax backported up to Thud! Update to latest dot release (or commit if EOL) in Yocto!
  - scripts/contrib/convert-overrides.py in poky to help you (not automagic!)
- More explanations on ??= operator

No-surprise operators

No-surprise operators

- =, +=, =+

```
GUESTS = "jasmine"
GUESTS += "christian"
```

```
GUESTS = "yao"
GUESTS =+ "marie"
```

```
GUESTS = "ashanti"
GUESTS += "muhammad"
GUESTS = "ilana"
```

No-surprise operators

- =, +=, =+

```
GUESTS = "jasmine"
GUESTS += "christian"
# Result: GUESTS => "jasmine christian"
```

```
GUESTS = "yao"
GUESTS =+ "marie"
# Result: GUESTS => "marie yao"
```

```
GUESTS = "ashanti"
GUESTS += "muhammad"
GUESTS = "ilana"
# Result: GUESTS => "ilana"
```

No-surprise operators

- `.=, =.`

```
MARK = "B"
MARK .= "+"
```

```
MONEY = "524,00$"
MONEY =. "1000000"
```

```
MARK = "A"
MARK .= "+"
MARK = "B"
```

No-surprise operators

- .=, =.

```
MARK = "B"
MARK .= "+"
# Result: MARK => "B+"
```

```
MONEY = "524,00$"
MONEY =. "1000000"
# Result: MONEY => "1000000524,00$"
```

```
MARK = "A"
MARK .= "+"
MARK = "B"
# Result: MARK => "B"
```

No-surprise operators

- .=, =., +=, =+ on unset variable

```
MARK .= "+"
```

```
MARK += "B"
```

No-surprise operators

- .=, =., +=, =+ on unset variable

```
MARK .= "+"
# Result: MARK => "+"
```

```
MARK += "B"
# Result: MARK => " B"
```

Default value operators

Default value operator - ?=

```
BUFFET ?= "tacos croissant baklava sushi"
BUFFET += "curry"
```

```
BUFFET += "ice-cream"
BUFFET ?= "schnitzel blini"
```

Default value operator - ?=

```
BUFFET ?= "tacos croissant baklava sushi"
BUFFET += "curry"
# Result: BUFFET => "tacos croissant baklava sushi curry"
```

```
BUFFET += "ice-cream"
BUFFET ?= "schnitzel blini"
# Result: BUFFET => "ice-cream"
```

Weak default value operator - ??=

- ??= is evaluated after all other operators

```
BUFFET ??= "tacos croissant baklava sushi"
BUFFET += "curry"
```

```
BUFFET += "ice-cream"
BUFFET ??= "schnitzel blini"
```

Weak default value operator - ??=

- ??= is evaluated after all other operators

```
BUFFET ??= "tacos croissant baklava sushi"
BUFFET += "curry"
# Result: BUFFET => "curry"
```

```
BUFFET += "ice-cream"
BUFFET ??= "schnitzel blini"
# Result: BUFFET => "ice-cream"
```

Weak default value operator - ??=

- ??= is evaluated after all other operators

```
BUFFET ??= "lokum gyoza hummus"
BUFFET ?= "pizza"
```

Weak default value operator - ??=

- ??= is evaluated after all other operators

```
BUFFET ??= "lokum gyoza hummus"
BUFFET ?= "pizza"
# Result: BUFFET => "pizza"
```

Weak default value operator - ??=

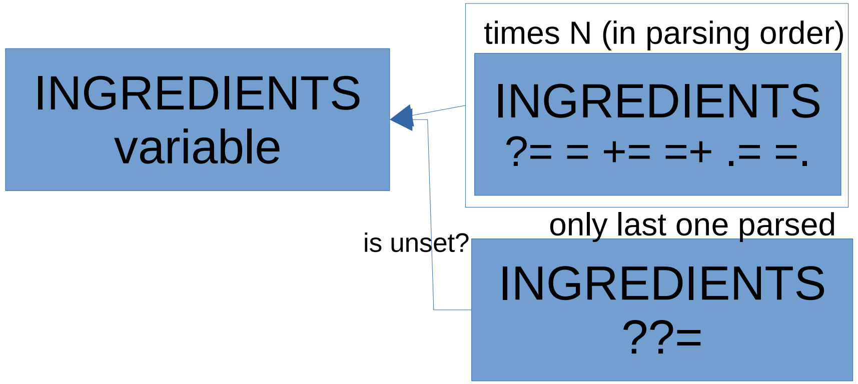
- ??= is evaluated after all other operators

```
BUFFET ??= "lokum gyoza hummus"
BUFFET ??= "pizza"
```

Weak default value operator - ??=

- ??= is evaluated after all other operators… and only the last parsed ??= is used!

```
BUFFET ??= "lokum gyoza hummus"
BUFFET ??= "pizza"
# Result: BUFFET => "pizza"
```



Non-obvious order

Non-obvious order

```
# cinnamon-buns recipe
INGREDIENTS ?= "cinnamon flour butter eggs milk sugar"
```

```
# local.conf
INGREDIENTS += "glaze vanilla-extract"
```

Non-obvious order

```
# cinnamon-buns recipe
INGREDIENTS ?= "cinnamon flour butter eggs milk sugar"
```

```
# local.conf
INGREDIENTS += "glaze vanilla-extract"
```

```
# Result: INGREDIENTS => "glaze vanilla-extract"
```

Debugging technique

Debugging technique

```
qschulz> bitbake-getvar -r cinnamon-buns INGREDIENTS
# $INGREDIENTS [2 operations]
#   append /yocto/build/conf/local.conf:268
#     "glaze vanilla-extract"
#   set? /yocto/meta-yp-summit/recipes-example/cinnamon-
buns/cinnamon-buns.bb:39
#     "cinnamon flour butter eggs milk sugar"
# pre-expansion value:
#   "glaze vanilla-extract"
INGREDIENTS="glaze vanilla-extract"
```

Debugging technique – pre-Honister (3.4)

```
qschulz> bitbake -e cinnamon-buns | \
          awk '/^# \$INGREDIENTS \[/,/^INGREDIENTS/'
# $INGREDIENTS [2 operations]
#   append /yocto/build/conf/local.conf:268
#     "glaze vanilla-extract"
#   set? /yocto/meta-yp-summit/recipes-example/cinnamon-
buns/cinnamon-buns.bb:39
#     "cinnamon flour butter eggs milk sugar"
# pre-expansion value:
#   "glaze vanilla-extract"
INGREDIENTS="glaze vanilla-extract"
```

Source: https://theyoctojester.info/session_14/main.html Josef Holzmayr a.k.a. LetoThe2nd a.k.a. TheYoctoJester

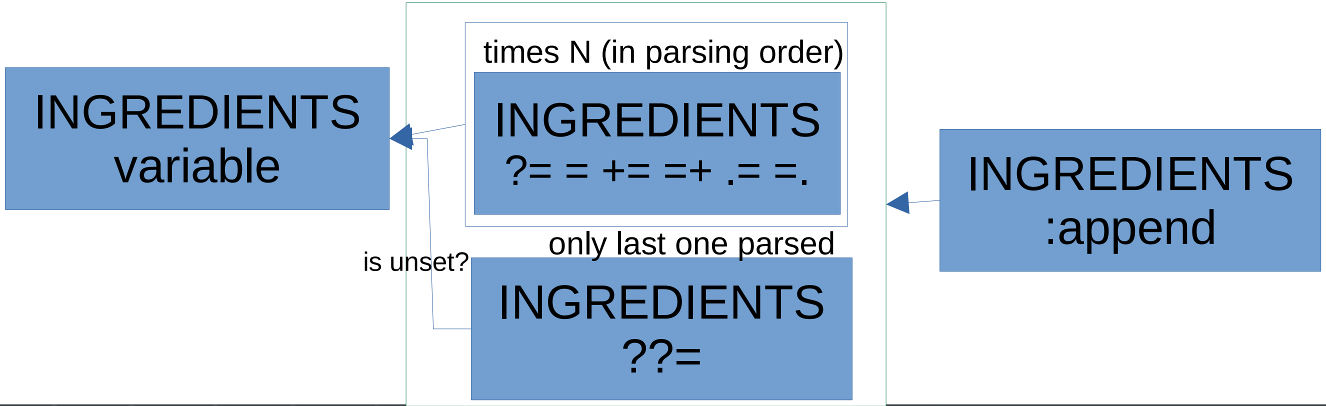
Debugging technique

- Configuration files (.conf) parsed before recipes, bbappends or include files
- New operator needed!

:append and :prepend

:append and :prepend

- Equivalent of a "postponed" .= and =.
- ~ post-its/pile
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then and only then :append/:prepend"



:append and :prepend

```
# cinnamon-buns recipe
INGREDIENTS ?= "cinnamon flour butter eggs milk sugar"
```

```
# local.conf
INGREDIENTS:append = " glaze vanilla-extract"
```

```
# Result: INGREDIENTS => "cinnamon flour butter eggs milk
sugar glaze vanilla-extract"
```

:append and :prepend

```
qschulz> bitbake-getvar -r cinnamon-buns INGREDIENTS
# $INGREDIENTS [2 operations]
#   :append /yocto/build/conf/local.conf:269
#     " glaze vanilla-extract"
#   set? /yocto/meta-yp-summit/recipes-example/cinnamon-
buns/cinnamon-buns.bb:42
#     "cinnamon flour butter eggs milk sugar"
# pre-expansion value:
#   "cinnamon flour butter eggs milk sugar glaze vanilla-
extract"
INGREDIENTS="cinnamon flour butter eggs milk sugar glaze
vanilla-extract"
```

Surprise, surprise

Surprise, surprise

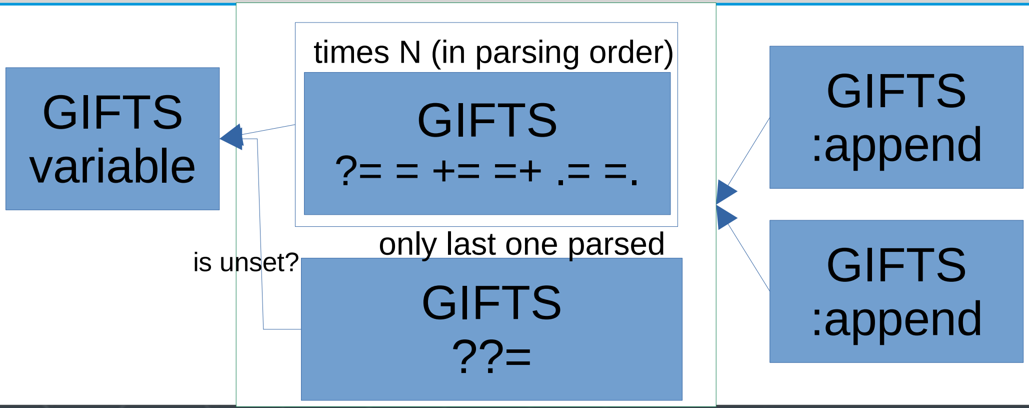
- ~~:append += => :append + leading space~~
  => removed in Kirkstone (4.0)!
- :append works on unset variable

```
# Unset GIFTS
GIFTS:append = " puzzle"
# Result: GIFTS => " puzzle"
```

Surprise, surprise

- **One :append (or :prepend) = one post-it/entry in the pile**

```
GIFTS = "train"
GIFTS:append = " puzzle"
# OOPSIES, my nephew wants a doll and not a puzzle
GIFTS:append = " doll"
```



Surprise, surprise

- One :append (or :prepend) = one post-it/entry in the pile

```
GIFTS = "train"
GIFTS:append = " puzzle"
# OOPSIES, my nephew wants a doll and not a puzzle
GIFTS:append = " doll"
# Result: GIFTS => "train puzzle doll"
```

- New operator!

Usage

Usage

- **:append/:prepend in .bbclass, .inc and .conf**

```
# cmake.bbclass
DEPENDS += "cmake-native"
# cmake-based-sw_0.bb
inherit cmake
DEPENDS = "bash"

# cmake-based-sw_1.bb
DEPENDS = "bash"
inherit cmake
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS += "cmake-native"
# cmake-based-sw_0.bb
inherit cmake
DEPENDS = "bash"
# Result: DEPENDS => "bash"
# cmake-based-sw_1.bb
DEPENDS = "bash"
inherit cmake
# Result: DEPENDS => "bash cmake-native"
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS += "cmake-native"
# boost.bbclass
DEPENDS = "boost"
# cmake-based-sw_2.bb
inherit cmake boost

# cmake-based-sw_3.bb
inherit boost cmake
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS += "cmake-native"
# boost.bbclass
DEPENDS = "boost"
# cmake-based-sw_2.bb
inherit cmake boost
# Result: DEPENDS => "boost"
# cmake-based-sw_3.bb
inherit boost cmake
# Result: DEPENDS => "boost cmake-native"
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS:append = " cmake-native"
# cmake-based-sw_0.bb
inherit cmake
DEPENDS = "bash"
# cmake-based-sw_1.bb
DEPENDS = "bash"
inherit cmake
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS:append = " cmake-native"
# cmake-based-sw_0.bb
inherit cmake
DEPENDS = "bash"
# cmake-based-sw_1.bb
DEPENDS = "bash"
inherit cmake

# Result for both: DEPENDS => "bash cmake-native"
```

Usage

- **:append/:prepend in .bbclass, .inc and .conf**

```
# cmake.bbclass
DEPENDS:append = " cmake-native"
# boost.bbclass
DEPENDS:append = " boost"
# cmake-based-sw_2.bb
inherit cmake boost

# cmake-based-sw_3.bb
inherit boost cmake
```

Usage

- :append/:prepend in .bbclass, .inc and .conf

```
# cmake.bbclass
DEPENDS:append = " cmake-native"
# boost.bbclass
DEPENDS:append = " boost"
# cmake-based-sw_2.bb
inherit cmake boost
# Result: DEPENDS => " cmake-native boost"
# cmake-based-sw_3.bb
inherit boost cmake
# Result: DEPENDS => " boost cmake-native"
```

Usage

- :append/:prepend in .bbclass, .inc and .conf
- For "context-specific" additions
- Should be avoided (hard to override)
- Best practice in recipes:

  1) inherit some-class

  2) Set/modify/append variables

Last word

Last word

- :remove, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then :append/:prepend then :remove"
- Final and removes all occurrences!

```
GIFTS = "train"
GIFTS:append = " puzzle"
GIFTS:append = " doll"
GIFTS:remove = "puzzle"
```

Last word

- :remove, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then :append/:prepend then :remove"
- Final and removes all occurrences!

```
GIFTS = "train"
GIFTS:append = " puzzle"
GIFTS:append = " doll"
GIFTS:remove = "puzzle"
# Result: GIFTS => "train  doll"
```

Last word

- :remove, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then :append/:prepend then :remove"
- Final and removes all occurrences!

```
GIFTS = "train"
GIFTS:append = " puzzle"
GIFTS:append = " doll"
GIFTS:remove = "puzzle"
# OOPSIES, my nephew actually wants a puzzle too
GIFTS:append = " puzzle"
```

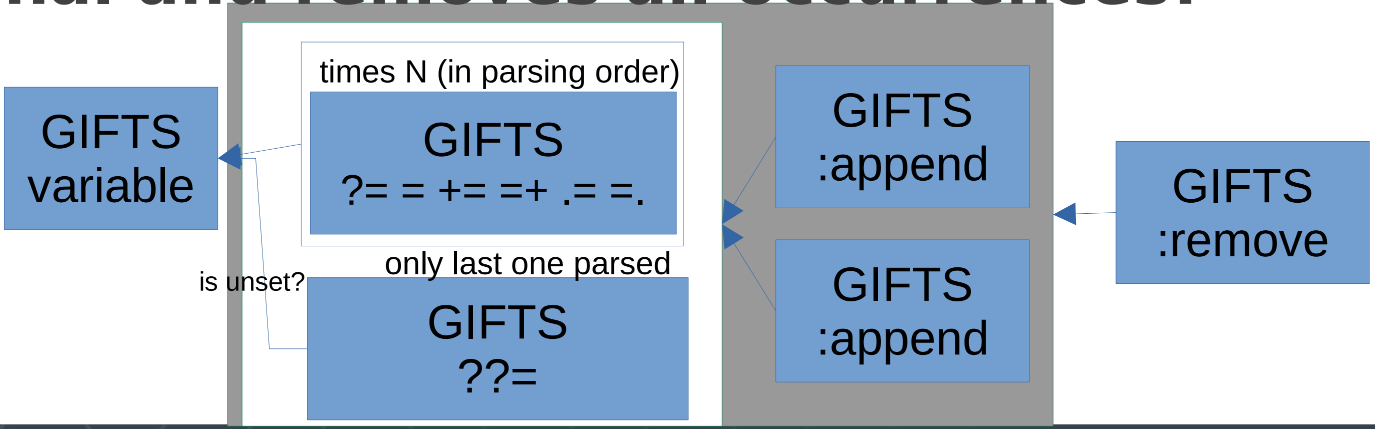
Last word

- :remove, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then :append/:prepend then :remove"
- Final and removes all occurrences!

```
GIFTS = "train"
GIFTS:append = " puzzle"
GIFTS:append = " doll"
GIFTS:remove = "puzzle"
# OOPSIES, my nephew actually wants a puzzle too
GIFTS:append = " puzzle"
# Result: GIFTS => "train  doll"
```

Last word

- :remove, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += =+ .= =. then :append/:prepend then :remove"
- Final and removes all occurrences!



Last word

- :remove, to be avoided (3rd party layer)
- :append/:prepend to be carefully chosen
- Trick to undo :remove

```
GIFTS = "train"
GIFTS:append = " puzzle"
GIFTS:append = " doll"
UNWANTED_GIFTS = "puzzle"
GIFTS:remove = "${UNWANTED_GIFTS}"
UNWANTED_GIFTS = ""
# Result: GIFTS => "train puzzle doll"
```

Source: https://theyoctojester.info/session_14/main.html Chris Larson

"Context-specific" operators

"Context-specific" operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
```

"Context-specific" operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For Elena
MORNING = "breakfast school nap play"
```

"Context-specific" operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For Elena
MORNING:elena = "breakfast school nap play"
```

- :elena operator = new pile for MORNING
- Overrides MORNING only for Elena after all ??= ?= = += =+ .= =., before :append
- MORNING:elena resolved (but unused!) even for Omar

"Context-specific" operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For Elena
MORNING:elena = "breakfast school nap play"
# For Esteban
MORNING:esteban = "breakfast school nap play"
```

- :elena and :esteban = MACHINE in Yocto

Avoiding copy-pasting

Avoiding copy-pasting

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For children (Elena, Esteban, ...)
MORNING:child = "breakfast school nap play"
```

- **:child = MACHINEOVERRIDES in Yocto**
- **Elena and Esteban are children, MORNING overridden for them**

Avoiding copy-pasting

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For children (Elena, Esteban, ...)
MORNING:child = "breakfast school nap play"
# For Elena
MORNING:elena = "breakfast school nap play hindi-language"
```

- **Individual > "kind"**
  MACHINE > MACHINEOVERRIDES in Yocto

Avoiding copy-pasting

```
MACHINEOVERRIDES = "kind1:kind2:kind3:kind4:${MACHINE}"
MACHINEOVERRIDES = "child:elena"
MACHINEOVERRIDES = "child:esteban"
MACHINEOVERRIDES = "omar"
```

- **Rightmost > leftmost**
- **:elena > :child**
- **:esteban > :child**

Combining operators

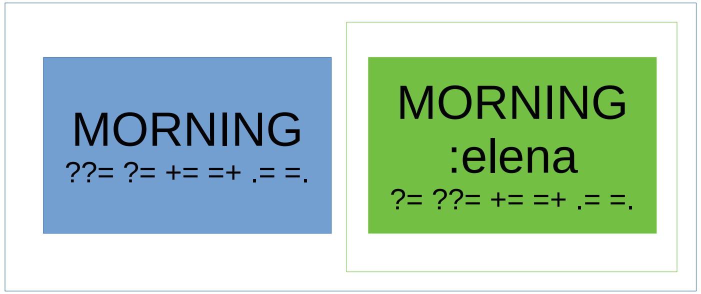
Combining operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For children (Elena, Esteban, ...)
MORNING:child = "breakfast school nap play"
MORNING += "play-with-dog"
```

Combining operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For children (Elena, Esteban, ...)
MORNING:child = "breakfast school nap play"
MORNING += "play-with-dog"
# Result for children: MORNING => "breakfast school nap
play"
# Result for Omar: MORNING => "make-breakfast breakfast
work sport play-with-dog"
```

Combining operators



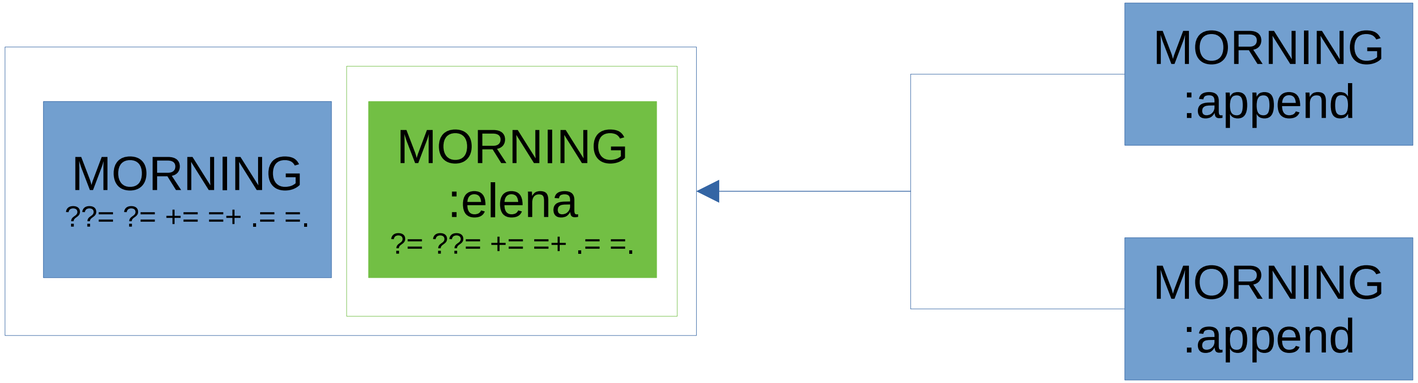
1. MORNING to be picked according to appropriate overrides

Combining operators

```
# For Omar
MORNING = "make-breakfast breakfast work sport"
# For children (Elena, Esteban, ...)
MORNING:child = "breakfast school nap play"
MORNING:append = " play-with-dog"
```

- :append after "normal" resolution
- For children, base MORNING = MORNING:child, then :append

Combining operators



2. MORNING:append to be applied

1. MORNING to be picked
according to appropriate overrides

Combining operators

```
# For all
MORNING = "breakfast"
# For children (Elena, Esteban, ...)
MORNING:child:append = " school nap play"
# For Omar
MORNING:omar:append = " make-breakfast work sport"
```

Combining operators

```
# For all
MORNING = "breakfast"
# For children (Elena, Esteban, ...)
MORNING:child:append = " school nap play"
# For Omar
MORNING:omar:append = " make-breakfast work sport"
# Result for children: MORNING => " school nap play"
# Result for Omar: MORNING => " make-breakfast work sport"
# Result for others: MORNING => "breakfast"
```

Reading combined operators

Reading combined operators

```
MORNING:omar:append
```

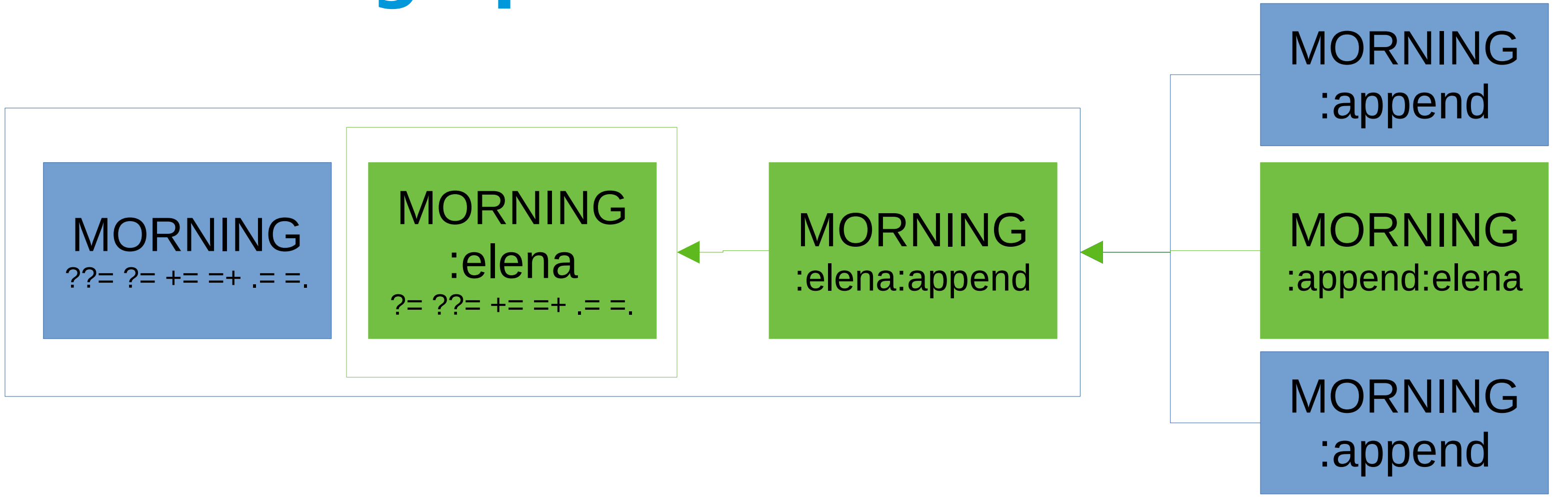
- **:omar = pile for Omar**
- **:append = pile for later (post-it)**
- **:omar:append creates a post-it for Omar-reserved pile**
- **Omar: MORNING = MORNING:omar, resolved (+= .= ...), then :append**

Reading combined operators

```
MORNING:append:omar
```

- :append = pile for later (post-it)
- :omar = only for Omar
- :append:omar creates for Omar only a post-it on top of default pile
- MORNING will be resolved, then :append:omar will be added if Omar

Combining operators



1. MORNING to be picked according to appropriate overrides

2. MORNING:append to be applied (default + appropriate overrides)

Reading combined operators

```
# For all
MORNING = "breakfast"
# For children (Elena, Esteban, ...)
MORNING:append:child = " school nap play"
# For Omar
MORNING:append:omar = " make-breakfast work sport"
```

Reading combined operators

```
# For all
MORNING = "breakfast"
# For children (Elena, Esteban, ...)
MORNING:append:child = " school nap play"
# For Omar
MORNING:append:omar = " make-breakfast work sport"
# Result for children: MORNING => "breakfast school nap
play"
# Result for Omar: MORNING => "breakfast make-breakfast
work sport"
# Result for others: MORNING => "breakfast"
```

Reading combined operators

- **When in doubt, bitbake-getvar!**

Now!

Now!

- := immediate expansion operator

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"
```

- FILESEXTRAPATHS:prepend
- THISDIR is directory at expansion time
  - Variables expanded after all parsing
  - Called "flattening"
  - If =, THISDIR is directory of recipe
  - If :=, THISDIR is dir of currently parsed file

Hidden use of operators

Immediate expansion by directives

- Applies to inherit, include and require

```
PACKAGECONFIG ?= “conf1”
inherit ${@bb.utils.contains('PACKAGECONFIG', 'conf1', 'conf1', '', d)}

# in a bbappend
PACKAGECONFIG:remove = “conf1”
```

Immediate expansion by directives

- Applies to inherit, include and require

```
PACKAGECONFIG ?= “conf1”
inherit ${@bb.utils.contains('PACKAGECONFIG', 'conf1', 'conf1', '', d)}

# in a bbappend
PACKAGECONFIG:remove = “conf1”
```

```
PACKAGECONFIG ?= “conf1”
inherit conf1 # inherited!

# in a bbappend
PACKAGECONFIG:remove = “conf1”
# PACKAGECONFIG = ”” in the end
```

OVERRIDES for files

```
# my-kernel.bb
SRC_URI += "file://defconfig"
```

```
meta-yp-summit
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   ├── aarch64
        │   │   └── defconfig <--- aarch64 machines
        │   └── defconfig <--- all other machines
        └── my-kernel_5.10.bb
```

OVERRIDES for files - gotchas

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── defconfig <--- want for my-aarch64-machine
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig
        └── my-kernel_5.10.bb
```

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── defconfig
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig <--- aarch64 machines
        └── my-kernel_5.10.bb
```

OVERRIDES for files - gotchas

- Fetch order from FILESPATH
- Check value with
  - bitbake-getvar -r my-kernel FILESPATH
  - Content of ${WORKDIR}/temp/log.do_fetch

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── aarch64
│       │       └── defconfig <--- aarch64 machines
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig
        └── my-kernel_5.10.bb
```

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── my-aarch64-machine
│       │       └── defconfig <--- my-aarch64-machine
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig <--- aarch64 machines
        └── my-kernel_5.10.bb
```

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── src
│       │       └── fileA
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── src
        │       ├── fileA
        │       └── fileB
        └── my-kernel_5.10.bb
```

```
# my-kernel_5.10.bb
SRC_URI = "file://src/"

# my-kernel_5.10.bbappend
FILESEXTRAPATHS:prepend :=
"${THISDIR}/my-kernel:"
```

OVERRIDES for files - gotchas

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── src
│       │       └── fileA
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── src
        │       ├── fileA
        │       └── fileB
        └── my-kernel_5.10.bb
```

```
# my-kernel_5.10.bb
SRC_URI = "file://src/"

# my-kernel_5.10.bbappend
FILESEXTRAPATHS:prepend :=
"${THISDIR}/my-kernel:"

# Result: only fileA from
the bbappend will be
fetched
```

Other OVERRIDES

Other OVERRIDES

```
OVERRIDES = "${TARGET_OS}:${TRANSLATED_TARGET_ARCH}:pn-${PN}:
${MACHINEOVERRIDES}:${DISTROOVERRIDES}:${CLASSOVERRIDE}
${LIBCOVERRIDE}:forcevariable"
```

- :pn-<recipename>
  - only makes sense from .conf files (c.f. Yocto chant #1)
- :class-native, :class-target, ...
- :libc-musl (LIBCOVERRIDE tclibc-musl.inc)
- :poky (DISTROOVERRIDES)
- :forcevariable (precedence over all except multilib)
- :virtclass-multilib-lib32 (MULTILIBS = "multilib:lib32")
- :task-compile (task-specific changes, e.g. PARALLEL_MAKE; note missing do_)

Sources

- https://docs.yoctoproject.org/bitbake/bitbake-user-manual/bitbake-user-manual-metadata.html#syntax-and-operators
- https://theyoctojester.info/session_14/main.html
- https://docs.yoctoproject.org/migration-guides/migration-3.4.html #override-syntax-changes
- https://docs.yoctoproject.org/migration-guides/migration-4.0.html#append-prepend-in-combination-with-other-operators
- Me with years of trial and error :)

- OVERRIDES history:
  - https://lists.openembedded.org/g/openembedded-architecture/message/1204
  - https://lists.openembedded.org/g/openembedded-architecture/message/1208

This is the end

LAST_WORDS:conference-2022:yocto-summit:speaker:quentin = "Thank you for attending! Any questions?"

Bonus slides

Surprise, surprise

Surprise, surprise

- **:append and :prepend for shell functions and tasks!**

```
prepare:prepend() {
    take shopping-list
}
prepare() {
    take wallet
}
prepare:append() {
    take tote-bag
}
```

```
do_groceries() {
    buy bread
}
do_groceries:append() {
    buy butter
}
do_groceries:prepend() {
    prepare
}
```

Surprise, surprise

- **:append and :prepend for Bitbake-style python functions and tasks!**

```
python do_walk_dog:prepend() {
    leash_dog()
}

python do_walk_dog() {
    go_outside()
    take_dog()
}
```

```
python end_walk_dog() {
    clean_paws()
}

python end_walk_dog:append() {
    feed_dog()
}
```

Importance of naming

Importance of naming

- Package managers don’t like uppercase letters
  - Debian package naming convention: https://www.debian.org/doc/debian-policy/ch-controlfields.html#source
- Naming convention:
  - No underscore, or capital letters for:
    - Recipe and package names,
    - Machine names, (MACHINEOVERRIDES),
    - Distro names, (DISTROOVERRIDES)

Files OVERRIDES algorithm

Files OVERRIDES algorithm

```
FILESEXTRAPATHS:prepend := "${THISDIR}/${BPN}:"
```

- **FILESOVERRIDES = "…:${MACHINEOVERRIDES}:…"**
- **Iterate from rightmost to leftmost FILESOVERRIDES**
- **For each iteration, iterate from leftmost to rightmost SRCDIRS (illustrative variable)**

```
SRCDIRS = "${FILESEXTRAPATHS}:/dir/${BP}:/dir/${BPN}:/dir/files"
SRCDIRS =
"/dir2/my-kernel:/dir/my-kernel-5.10:/dir/my-kernel:/dir/files"
# /dir = /path/to/meta-yp-summit/recipes-kernel/linux
# /dir2 = /path/to/meta-yp-summit/recipes-append/linux
```

- **FILESPATH contains the list of paths to traverse**

Files OVERRIDES algorithm

```
# pseudo-code
file # file listed in SRC_URI
paths = ${FILESEXTRAPATHS} + /dir/${BP} + /dir/${BPN} + /dir/files

for path in $paths
    for override in reverse(${FILESOVERRIDES})
        if exists($path/$override/$file)
            return found

    if exists($path/$file)
        return found

error "file not found"
```

Files OVERRIDES algorithm

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── aarch64
│       │       └── defconfig <--- aarch64 machines
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig
        └── my-kernel_5.10.bb
```

Files OVERRIDES algorithm

```
meta-yp-summit
├── recipes-append
│   └── linux
│       ├── my-kernel
│       │   └── my-aarch64-machine
│       │       └── defconfig <--- my-aarch64-machine
│       └── my-kernel_5.10.bbappend
└── recipes-kernel
    └── linux
        ├── my-kernel
        │   └── aarch64
        │       └── defconfig <--- aarch64 machines
        └── my-kernel_5.10.bb
```

Python-equivalent operators

Python-equivalent operators

- **Meaningful within anonymous python functions => task scope!**

```
X = "A" => d.setVar("X", "A")
${X} => d.getVar("X")
Y := ${X} => d.getVar("X", True)
X:append = "B" => d.appendVar("X", "B")
X:prepend = "C" => d.prependVar("X", "C")
```

- ***Source: https://docs.yoctoproject.org/bitbake/bitbake-user-manual/bitbake-user-manual-metadata.html#functions-for-accessing-datastore-variables***

This is the (real) end

LAST_WORDS:conference-2022:yocto-summit:speaker:quentin = "Thank you for attending! Any questions?"

yocto PROJECT

THE LINUX FOUNDATION