Yocto Project®
Demystifying the OVERRIDES mechanism and Bitbake operators
Quentin Schulz, StreamUnlimited
Slides under CC BY-SA-4.0
Yocto Project Summit, May 2021
About

- BSP engineer @ StreamUnlimited GmbH
- Smart speakers
- SoM for anything related to audio or voice assistance
- StreamSDK SW stack:
  - Google Cast, Amazon Alexa, Spotify, Airplay,
    multiroom audio, Dolby Atmos, Multi-UI...
- Open positions:
  https://www.streamunlimited.com/vacancies/
- quentin.schulz [at] streamunlimited.com
- foss [at] 0leil.net (zero not o)
No-surprise operators
No-surprise operators

- =, +=, =+  

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

GUESTS = "aaron"
GUESTS += "marie"

GUESTS += "muhammad"
GUESTS = "ilana"
```
No-surprise operators

- \(=, +=, =+\)

\[
\begin{align*}
\text{GUESTS} &= "jasmine" \\
\text{GUESTS} &=+ "christian" \\
& \text{# Result: GUESTS }=> "jasmine christian"
\end{align*}
\]

\[
\begin{align*}
\text{GUESTS} &= "aaron" \\
\text{GUESTS} &=+ "marie" \\
& \text{# Result: GUESTS }=> "marie aaron"
\end{align*}
\]

\[
\begin{align*}
\text{GUESTS} &=+ "muhammad" \\
\text{GUESTS} &= "ilana" \\
& \text{# Result: GUESTS }=> "ilana"
\end{align*}
\]
No-surprise operators

• .=, =.

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

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

```plaintext
MARK .= "+
MARK = "B"
```
No-surprise operators

- `.=`,

\[
\begin{align*}
\text{MARK} &= "B" \\
\text{MARK} \ . &= "+" \\
\text{# Result: MARK} &\Rightarrow "B+"
\end{align*}
\]

\[
\begin{align*}
\text{MONEY} &= "524,00\$" \\
\text{MONEY} \ . &= "10000000" \\
\text{# Result: MONEY} &\Rightarrow "1000000524,00\$"
\end{align*}
\]

\[
\begin{align*}
\text{MARK} \ . &= "+" \\
\text{MARK} &= "B" \\
\text{# Result: MARK} &\Rightarrow "B"
\end{align*}
\]
Weak assignment operators
Weak assignment operators

- `?=, ??=`

```plaintext
BUFFET ?= "tacos croissant baklava sushi"
BUFFET += "curry"
BUFFET ??= "lokum gyoza hummus"
BUFFET ?= "pizza"
BUFFET += "paella"
BUFFET += "ice-cream"
BUFFET ?= "schnitzel blini"
```
Weak assignment operators

- `?=`, `??=`

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

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

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

```plaintext
# cinnamon-buns recipe
INGREDIENTS += "cinnamon flour yeast butter eggs milk sugar"

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

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

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

# Result: INGREDIENTS => "glaze vanilla-extract"
Debugging technique

```
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 yeast 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 n°1

• 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/procrastinator
- Tell Bitbake "resolve ??= ?? = = += += .= =. then and only then _append/_prepend"

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

```bash
# local.conf
INGREDIENTS += "glaze vanilla-extract"
```
_append and _prepend

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

# local.conf
INGREDIENTS_append = " glaze vanilla-extract"

# Result: INGREDIENTS => "cinnamon flour yeast butter eggs milk sugar glaze vanilla-extract"
append and prepend

```bash
qschulz> bitbake -e cinnamon-buns | \
        awk '/^# \$INGREDIENTS \[/,/^INGREDIENTS/'
# $INGREDIENTS2 [2 operations]
# _append /yocto/build/conf/local.conf:269
#   " glaze vanilla-extract"
# set? /yocto/metal-yp-summit/recipes-example/cinnamon-buns/cinnamon-buns.bb:42
#   "cinnamon flour yeast butter eggs milk sugar"
# pre-expansion value:
#   "cinnamon flour yeast butter eggs milk sugar glaze vanilla-extract"
INGREDIENTS="cinnamon flour yeast butter eggs milk sugar glaze vanilla-extract"
```
Surprise, surprise bis
Surprise, surprise bis

- `_append += => _append + leading space`
- One `_append` (or `_prepend`) = one post-it/entry in the pile

```python
GIFTS = "train"
GIFTS_append = " puzzle"
# OOPSIES, my nephew wants a doll and not a puzzle
GIFTS_append = " doll"
```
Surprise, surprise bis

- `_append += => _append + leading space`
- 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!
Last word

- `_remove`, another "postponed" operator
- Tell Bitbake "resolve ??= ?= = += == .= =. then _append/_prepend then _remove"
- Final and removes all occurrences!

```plaintext
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`, to be avoided (3rd party layer)
- `_append/_prepend` to be carefully chosen
- Trick to undo `_remove`

```python
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
Now!

• := immediate expansion operator
  FILESEXTAPATHS_prepend := "${THISDIR}/files:"

• FILESEXTAPATHS_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
Usage

- `_append/_prepend in `.bbclass`, `.inc` and `.conf`

```bash
# 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`

```bash
# 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

```bash
# 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`

```bash
# 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`

```bash
# 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

```bash
# 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`

```bash
# 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

```plaintext
# 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
"Context-specific" operators
"Context-specific" operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport make-dinner dinner sleep"
"Context-specific" operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport
make-dinner dinner sleep"

# For Elena
TODAY_TODO = "breakfast school nap play dinner sleep"
"Context-specific" operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport
make-dinner dinner sleep"

# For Elena
TODAY_TODO_elena = "breakfast school nap play dinner sleep"

- _elena operator = new pile for TODAY_TODO
- Overrides TODAY_TODO only for Elena after all ??= ?= = += == .= =.., before _append
- TODAY_TODO_elena resolved (but unused!) even for Omar
"Context-specific" operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport make-dinner dinner sleep"

# For Elena
TODAY_TODO_elena = "breakfast school nap play dinner sleep"

# For Esteban
TODAY_TODO_esteban = "breakfast school nap play dinner sleep"

• _elena and _esteban = MACHINE in Yocto
Avoiding copy-pasting
Avoiding copy-pasting

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport make-dinner dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_child = "breakfast school nap play dinner sleep"

• _child = MACHINEOVERRIDES in Yocto
• Elena and Esteban are children, TODAYTODO overridden for them
Avoiding copy-pasting

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport make-dinner dinner sleep"
# For children (Elena, Esteban, ...)
TODAY_TODO_child = "breakfast school nap play dinner sleep"
# For Elena
TODAY_TODO_elena = "breakfast school nap play dinner sleep 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
TODAY_TODO = "make-breakfast breakfast commute work sport
make-dinner dinner sleep"
# For children (Elena, Esteban, ...)
TODAY_TODO_child = "breakfast school nap play dinner sleep"
TODAY_TODO += "play-with-dog"
Combining operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport make-dinner dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_child = "breakfast school nap play dinner sleep"

TODAY_TODO += "play-with-dog"

# Result for children: TODAY_TODO => "breakfast school nap play dinner sleep"
# Result for Omar: TODAY_TODO => "make-breakfast breakfast commute work sport make-dinner dinner sleep play-with-dog"
Combining operators

# For Omar
TODAY_TODO = "make-breakfast breakfast commute work sport
make-dinner dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_child = "breakfast school nap play dinner sleep"
TODAY_TODO_append = " play-with-dog"

- _append after "normal" resolution
- For children, base TODAY_TODO = TODAY_TODO_child, then _append
Combining operators

# For all
TODAY_TODO = "breakfast dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_child_append = " school nap play"

# For Omar
TODAY_TODO_omar_append = " make-breakfast make-dinner commute work sport"
Combining operators

# For all
TODAY_TODO = "breakfast dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_child_append = " school nap play"

# For Omar
TODAY_TODO_omar_append = " make-breakfast make-dinner commute work sport"

# Result for children: TODAY_TODO => " school nap play"
# Result for Omar: TODAY_TODO => " make-breakfast make-dinner commute work sport"
# Result for others: TODAY_TODO => "breakfast dinner sleep"
Combining operators

```python
# For all
TODAY_TODO = "breakfast dinner sleep"
# For children (Elena, Esteban, ...)
TODAY_TODO_append_child = " school nap play"
# For Omar
TODAY_TODO_append_omar = " make-breakfast make-dinner commute work sport"
```
Combining operators

# For all
TODAY_TODO = "breakfast dinner sleep"

# For children (Elena, Esteban, ...)
TODAY_TODO_append_child = " school nap play"

# For Omar
TODAY_TODO_append_omar = " make-breakfast make-dinner commute work sport"

# Result for children: TODAY_TODO => "breakfast dinner sleep school nap play"
# Result for Omar: TODAY_TODO => "breakfast dinner sleep make-breakfast make-dinner commute work sport"
# Result for others: TODAY_TODO => "breakfast dinner sleep"
Reading combined operators
Reading combined operators

\[
\text{TODAY\_TODO\_omar\_append}
\]

- \_omar = pile for Omar
- \_append = pile for later (post-it)
- \_omar\_append creates a post-it for Omar-reserved pile
- Omar: TODAY\_TODO = TODAY\_TODO\_omar, resolved (+= .= ...), then \_append
Reading combined operators

**TODAY_TODO_append_omar**

- `_append = pile for later (post-it)`
- `_omar = pile for Omar`
- `_append_omar creates for Omar only a post-it on top of default pile`
- TODAY_TODO will be resolved, then `_append_omar` will be added if Omar
Reading combined operators

\[
\text{TODAY\_TODO} = \text{"breakfast dinner sleep"}
\]
\[
\text{TODAY\_TODO\_append} = \text{" play-with-dog"}
\]

\# For children (Elena, Esteban, ...)
\[
\text{TODAY\_TODO\_append\_child} = \text{" school nap play"}
\]

\# For Omar
\[
\text{TODAY\_TODO\_append\_omar} = \text{" make-breakfast make-dinner commute work sport"}
\]

\# Result for children: TODAY\_TODO => "breakfast dinner sleep play-with-dog school nap play"
\# Result for Omar: TODAY\_TODO => "breakfast dinner sleep play-with-dog make-breakfast make-dinner commute work sport"
\# Result for others: TODAY\_TODO => "breakfast dinner sleep play-with-dog"
Reading combined operators

• When in doubt, bitbake -e!
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 my-kernel -e | awk '/^# \$FILESPATH /[/,/^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

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

# my-kernel_5.10.bbappend
FILESEXTRAPATHS_prepend := "${THISDIR}/my-kernel:")
```
OVERRIDES for files - gotchas

```
# 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

OVERRIDES = "${TARGET_OS}:${TRANSLATED_TARGET_ARCH}:pn-${PN}:${MACHINEOVERRIDEST}:${DISTROOVERRIDEST}:${CLASSOVERRIDEST}:${LIBCOVERRIDEST}:forcevariable"

- _pn-<recipename>
  - only makes sense from .conf files (c.f. Yocto chant #1)
- _class-native, class-target, ...
- _virtclass-multilib-lib32 (MULTILIBS = "multilib:lib32")
- _libc-musl (LIBCOVERRIDEST tclibc-musl.inc)
- _poky (DISTROOVERRIDEST)
- _forcevariable (takes precedence over everything)
Other OVERRIDES

- _2.0, _3.2+ (BBVERSIONS; rarely used)
- _task-compile (task-specific changes, e.g. PARALLEL_MAKE; note missing do_)
- Probably others :)
Sources

- https://theyoctojester.info/session_14/main.html
- 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
  - What’s next for OVERRIDES? Not much :)

78
LAST_WORDS_conference_2021_yocto-summit_speaker_quentin = "Thank you for attending! Any questions?"
Bonus slides
Surprise, surprise
**Surprise, surprise**

- `_append` and `_prepend` for shell functions and tasks!

```bash
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()
}

do_walk_dog() {
    go_outside()
    take_dog()
}

do_end_walk_dog-prepend() {
    clean_paws()
}

do_end_walk_dog() {
    feed_dog()
}
```
Importance of naming
Importance of naming

- **TODAY_TODO_omar_append**
- Variables:
  - All uppercase,
  - Underscore possible,
- Overrides:
  - Prefixed with an underscore,
- String from variable or override?
  - Override:
    - No underscore,
    - All lowercase,
Importance of naming

• Package managers don’t like uppercase letters
  • Debian package naming convention:

• 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 = "...:${MACHINEOVERRIDERS}:...
• Iterate from rightmost to leftmost FILESOVERRIDERS
• 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:

This is the (real) end

LAST_WORDS_conference_2021_yocto-summit_speaker_quentin = "Thank you for attending! Any questions?"