

VCPKG IS A GREAT TOOL

Martin Hořeňovský

PEX

VCPKG IS A GREAT TOOL

Martin Hořeňovský



no, seriously, it is great ...

no, seriously, it is great ...
... in the context of C++ package managers

BUT

It is a Microsoft product through and through

It is a Microsoft product through and through
backwards compatibility is more important than fixing
past mistakes

Let's talk about the "fun" issues you can run into

INCONSISTENT VERSION RESOLUTION

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg",
  "name": "boost-mismatch",
  "builtin-baseline": "d090b933e923c0a69950423ae81fb9488d2d7bf",
  "dependencies": [
    {
      "name": "boost-circular-buffer",
      "version>=": "1.80.0"
    }
  ]
}
```

```
boost-mismatch> cmake -B build -S . ^  
                -DCMAKE_TOOLCHAIN_FILE=c:/ubuntu/vcpkg/scripts/bui  
  
-- Running vcpkg install  
Detecting compiler hash for triplet x64-windows...  
The following packages will be built and installed:  
* boost-assert:x64-windows -> 1.79.0  
  boost-circular-buffer:x64-windows -> 1.80.0  
* boost-concept-check:x64-windows -> 1.79.0  
* boost-config:x64-windows -> 1.79.0  
* boost-core:x64-windows -> 1.79.0  
...  
...
```

I originally found this issue with Boost packages,

I originally found this issue with Boost packages,
and I got it partially fixed for Boost 1.80#1.

I originally found this issue with Boost packages,
and I got it partially fixed for Boost 1.80#1.
At least for the trivial case.

Boost packages now use version constraints when referencing other Boost packages

Boost packages now use version constraints when
referencing other Boost packages

It is still easy to break by mistake though


```
{  
  "name": "boost-mismatch-2",  
  "builtin-baseline": "d090b933e923c0a69950423ae81fb9488d2d7bf",  
  "dependencies": [  
    { "name": "boost-circular-buffer", "version>=": "1.81.0" },  
    "some-lib",  
  ]  
}
```

```
{
  "name": "boost-mismatch-2",
  "builtin-baseline": "d090b933e923c0a69950423ae81fb9488d2d7bf",
  "dependencies": [
    { "name": "boost-circular-buffer", "version>=": "1.81.0" },
    "some-lib",
  ]
}
```

```
{
  "name": "some-lib",
  "dependencies": [
    { "name": "boost-icl", "version>=": "1.82.0" }
  ]
}
```

```
boost-mismatch-2> cmake -B build -S . ^
                    -DCMAKE_TOOLCHAIN_FILE=c:/ubuntu/vcpkg/scripts/

-- Running vcpkg install
Detecting compiler hash for triplet x64-windows...
The following packages will be built and installed:
    ...
* boost-bind:x64-windows -> 1.82.0
* boost-build:x64-windows -> 1.82.0
  boost-circular-buffer:x64-windows -> 1.81.0
    ...
```

This issue applies to any "split package" of monolithic project

This issue applies to any "split package" of monolithic project

e.g. Qt

OVERBUILDING TRANSITIVE DEPENDENCIES

vcpkg has a (mis)feature called "default features"

vcpkg has a (mis)feature called "default features"
This is a set of features of port to build by default

Default features often lead to large package

Default features often lead to large package

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg",
  "name": "libfoo",
  "dependencies": [
    {
      "name": "opencv4",
      "features": ["png"],
      "version>=": "4.8.0"
    }
  ]
}
```

```
libfoo$ cmake -B build -S . \  
    -DCMAKE_TOOLCHAIN_FILE=~/.vcpkg/scripts/buildsystems/vcpkg.  
  
-- Running vcpkg install  
Detecting compiler hash for triplet x64-linux...  
The following packages will be built and installed:  
* at-spi2-atk:x64-linux -> 2.38.0  
* at-spi2-core:x64-linux -> 2.44.1#2  
* atk:x64-linux -> 2.38.0#5  
* brotli:x64-linux -> 1.0.9#5  
* bzip2[core,tool]:x64-linux -> 1.0.8#4  
* cairo[core,fontconfig,freetype,gobject,x11]:x64-linux -> 1  
* dirent:x64-linux -> 1.23.2#2  
* egl-registry:x64-linux -> 2022-09-20  
* expat:x64-linux -> 2.5.0#3  
* fontconfig:x64-linux -> 2.14.2  
* freetype[brotli,bzip2,core,png,zlib]:x64-linux -> 2.12.1#3  
...  
Installing 1/44 vcpkg-cmake-config:x64-linux...
```

Thankfully you can disable them

Thankfully you can disable them

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg",
  "name": "libfoo",
  "dependencies": [
    {
      "name": "opencv4",
      "default-features": false,
      "features": ["png"],
      "version>=": "4.8.0"
    }
  ]
}
```

```
libfoo$ cmake -B build -S . \  
    -DCMAKE_TOOLCHAIN_FILE=~/.vcpkg/scripts/buildsystems/vcpkg.  
  
-- Running vcpkg install  
Detecting compiler hash for triplet x64-linux...  
The following packages will be built and installed:  
* libpng:x64-linux -> 1.6.39#1  
  opencv4[core,png]:x64-linux -> 4.8.0  
* vcpkg-cmake:x64-linux -> 2022-12-22  
* vcpkg-cmake-config:x64-linux -> 2022-02-06#1  
* vcpkg-get-python-packages:x64-linux -> 2022-06-30  
* zlib:x64-linux -> 1.2.13  
  
Installing 1/6 vcpkg-cmake:x64-linux...
```

But vcpkg can (and will) happily ignore that

```
{  
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpk  
  "name": "foosdk",  
  "dependencies": ["libfoo"]  
}
```



```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg",
  "name": "foosdk",
  "dependencies": ["libfoo"]
}
```

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg",
  "name": "libfoo",
  "dependencies": [
    {
      "name": "opencv4",
      "default-features": false,
      "features": ["png"],
      "version>=": "4.8.0"
    }
  ]
}
```

```
foosdk$ cmake -B build -S . -DCMAKE_TOOLCHAIN_FILE=~/.vcpkg/scr

-- Running vcpkg install
Detecting compiler hash for triplet x64-linux...
The following packages will be built and installed:
* at-spi2-atk:x64-linux -> 2.38.0
* at-spi2-core:x64-linux -> 2.44.1#2
* atk:x64-linux -> 2.38.0#5
* brotli:x64-linux -> 1.0.9#5
* bzip2[core,tool]:x64-linux -> 1.0.8#4
* cairo[core,fontconfig,freetype,gobject,x11]:x64-linux -> 1
* dirent:x64-linux -> 1.23.2#2
* egl-registry:x64-linux -> 2022-09-20
* expat:x64-linux -> 2.5.0#3
* fontconfig:x64-linux -> 2.14.2
* freetype[brotli,bzip2,core,png,zlib]:x64-linux -> 2.12.1#3
...

Installing 1/45 vcpkg-cmake-config:x64-linux...
```


vcpkg flat out ignores the "default-features" option in dependencies

vcpkg flat out ignores the "default-features" option in dependencies

A PR to fix this has been open for over 2 years

It can get worse than just building too much

It can get worse than just building too much
ffmpeg's default build used to be GPL licensed

 **@Xarn** This is completely unreasonable.



Neumann-A Today at 12:22 AM

No it is not unreasonable it is documented. If you don't care about all your dependencies vcpkg is free to install whatever it deems reasonable. If you want other behavior just be explicit about it.



Xarn ^{HE}_{HIM} Today at 12:23 AM

I can document that if you run test compiled with Catch2 and it finds vcpkg directory, it prints "lol, lmao" and formats your hard drive.

That won't make it reasonable.

CONFLICT IN VERSION CONSTRAINTS

vcpkg supports only `version >=` constraints

vcpkg supports only `version >=` constraints

But resolving two constraints can still fail

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg
"name": "foosdk",
"dependencies": [
  { "name": "abseil", "version>=": "20230802.1" },
  "libfoo"
]
}
```

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg
"name": "foosdk",
"dependencies": [
  { "name": "abseil", "version>=": "20230802.1" },
  "libfoo"
]
}
```

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg
"name": "libfoo",
"dependencies": [
  { "name": "abseil", "version>=": "20211102.1" }
]
}
```

```
foosdk> cmake -B build -S . ^
  -DCMAKE_TOOLCHAIN_FILE=C:\ubuntu\vcpkg\scripts\buildsystems

-- Running vcpkg install
Fetching registry information from https://github.com/Microsoft
error: version conflict on absl:x64-windows:
    foosdk required 20230802.1, which cannot be compared wi

The versions have incomparable schemes:
  absl@20211102.1 has scheme string
  absl@20230802.1 has scheme relaxed
```

It can happen with just one port, if the "baseline" has conflicting version scheme

It can happen with just one port, if the "baseline" has conflicting version scheme

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpk
  "name": "foosdk",
  "builtin-baseline": "826ebc235f28261dbf150fe558f6fc00b4783a3
  "dependencies": [
    { "name": "abseil", "version>=": "20230802.1" }
  ]
}
```

```
foosdk> cmake -B build -S . ^
  -DCMAKE_TOOLCHAIN_FILE=C:\ubuntu\vcpkg\scripts\buildsystems

-- Running vcpkg install
error: version conflict on abseil:x64-windows:
      foosdk required 20230802.1, which cannot be
      compared with the baseline version 20211102.1.

The versions have incomparable schemes:
  abseil@20211102.1 has scheme string
  abseil@20230802.1 has scheme relaxed
```


vcpkg recognizes 4 different version types

- version (relaxed version)
- version-semver
- version-date
- version-string

vcpkg only compares versions within the same domain

vcpkg only compares versions within the same domain
version-strings are incomparable by definition

In Jan 2022 I convinced Billy that version-semver and version-relaxed are the same domain

In Jan 2022 I convinced Billy that version-semver and
version-relaxed are the same domain

But vcpkg will never accept `2021.01.01` as a version

CONSTRAINTS OVERRIDEN BY BASELINE

The baseline always inserts constraints into resolution

The baseline always inserts constraints into resolution

```
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpk
  "name": "baseline-constraints",
  "builtin-baseline": "0e47c1985273129e4d0ee52ff73bed912555de
  "dependencies": [
    { "name": "fmt", "version>=": "9.1.0" }
  ]
}
```



```
baseline-constraints>cmake -B build -S . -DCMAKE_TOOLCHAIN_FILE
-- Building for: Visual Studio 16 2019
-- Running vcpkg install
Detecting compiler hash for triplet x64-windows...
The following packages will be built and installed:
  fmt:x64-windows -> 10.1.1
* vcpkg-cmake:x64-windows -> 2023-05-04
* vcpkg-cmake-config:x64-windows -> 2022-02-06#1
```

If you want your versions to have priority, you have to
use old baseline

If you want your versions to have priority, you have to
use old baseline

Using old baseline means that you use old versions of
transitive dependencies

The other option is using explicit overrides

The other option is using explicit overrides
But overrides don't propagate from dependencies

The other option is using explicit overrides

But overrides don't propagate from dependencies

This breaks the dependency management abstraction

CONCLUSION

vcpkg is functional package manager with lot of warts

vcpkg is functional package manager with lot of warts

It will be forever held back by the backwards
compatibility

vcpkg is functional package manager with lot of warts

It will be forever held back by the backwards
compatibility

Some of these issues can cause lot of problems

vcpkg is functional package manager with lot of warts

It will be forever held back by the backwards
compatibility

Some of these issues can cause lot of problems

And it is good to be aware of them

But vcpkg also gets lot of things right

Package metadata are non-executable

Package metadata are non-executable
It passes through the package build system

Package metadata are non-executable

It passes through the package build system

Curated set of packages patched and tested together

Package metadata are non-executable

It passes through the package build system

Curated set of packages patched and tested together

Generally easy to use

Overall vcpkg has served us well

THE END

