Как мы начали использовать vcpkg

и переехали на git

Денис Панин

Мастер на все руки @ NVIDIA

В чем проблема переехать на git?

```
#include "../../../../самодельные_куски_буста.h"
```

В чем проблема переехать на git?

```
#include "../../../../самодельные_куски_буста.h"

#pragma comment(lib, "../../cool_lib/vc12_only.lib")
```

В чем проблема переехать на git?

```
#include "../../../../самодельные_куски_буста.h"

#pragma comment(lib, "../../cool_lib/vc12_only.lib")
```

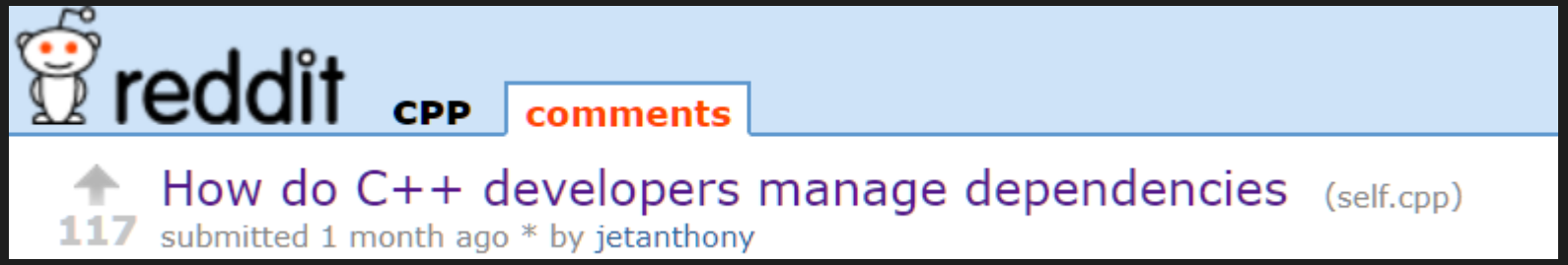
Общий код между компонентами

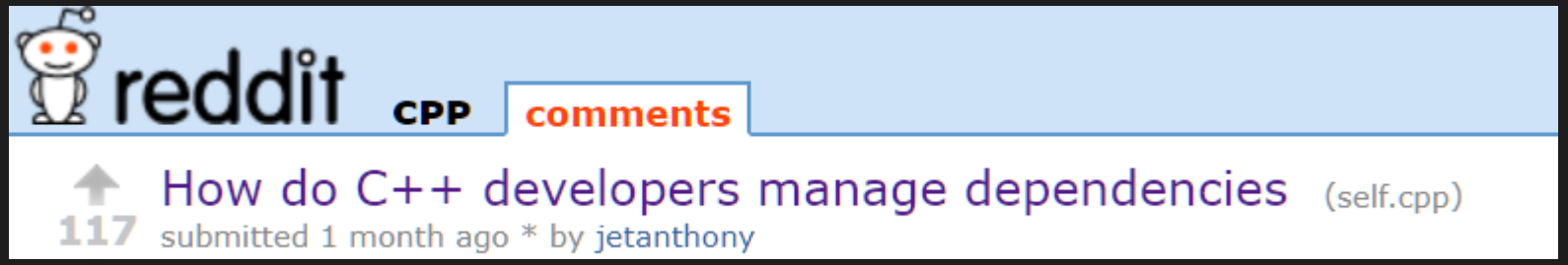
Menu | Perforce Software | Develo | Perforce Software, Inc. [US] | www.perforce.com

How NVIDIA Manages 600 Million Files

The leader in visual computing explains why they've standardized on Helix Core.

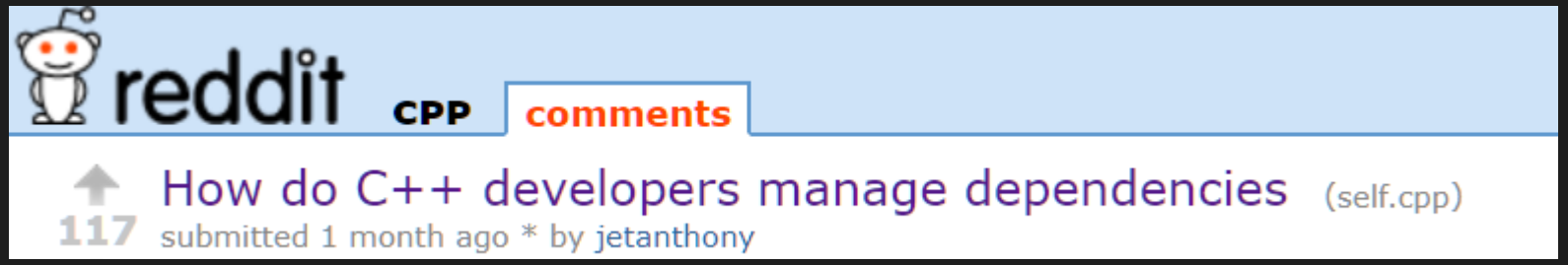
WATCH ALL VIDEO TESTIMONIALS


reddit
CPP
comments
How do C++ developers manage dependencies
(self.cpp)
117
submitted 1 month ago * by jetanthony



[–] **ooglesworth** 151 points 1 month ago

Through much pain and anguish.



[–] **ooglesworth** 151 points 1 month ago

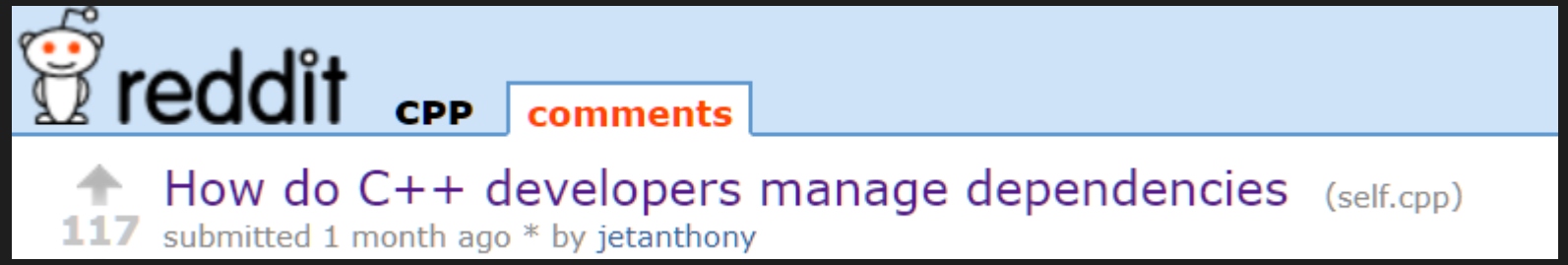
Through much pain and anguish.

[–] **quzox_** 46 points 1 month ago

Step 1: Buy a lot of vodka

Step 2: Drink it

Step 3: Hope that the build is OK



[–] **ooglesworth** 151 points 1 month ago

Through much pain and anguish.

[–] **Droggl** 22 points 1 month ago

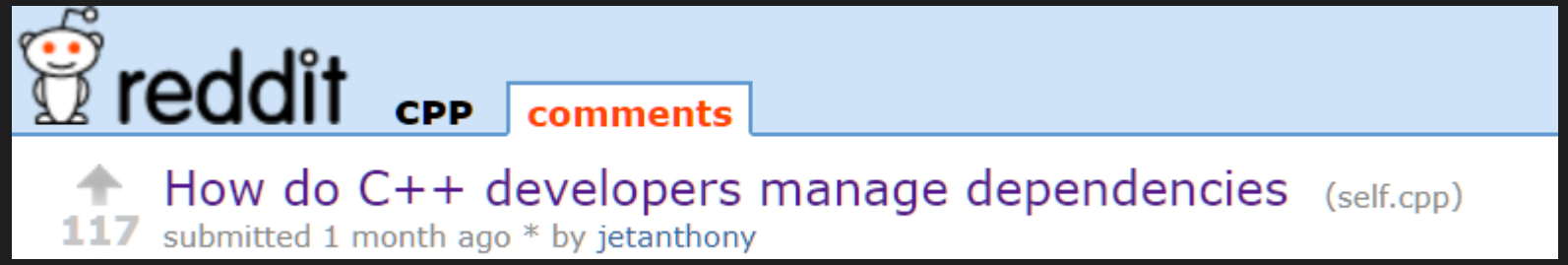
Its so painful in c++ that most devs i know rather reinvent any given functionality than import it from somewhere. I guess you could say c++ has built-in NIH support.

[–] **quzox_** 46 points 1 month ago

Step 1: Buy a lot of vodka

Step 2: Drink it

Step 3: Hope that the build is OK



[–] **ooglesworth** 151 points 1 month ago

Through much pain and anguish.

[–] **emthreeve** 7 points 1 month ago

From years of experience in several large companies: they don't, mostly.

[–] **Droggl** 22 points 1 month ago

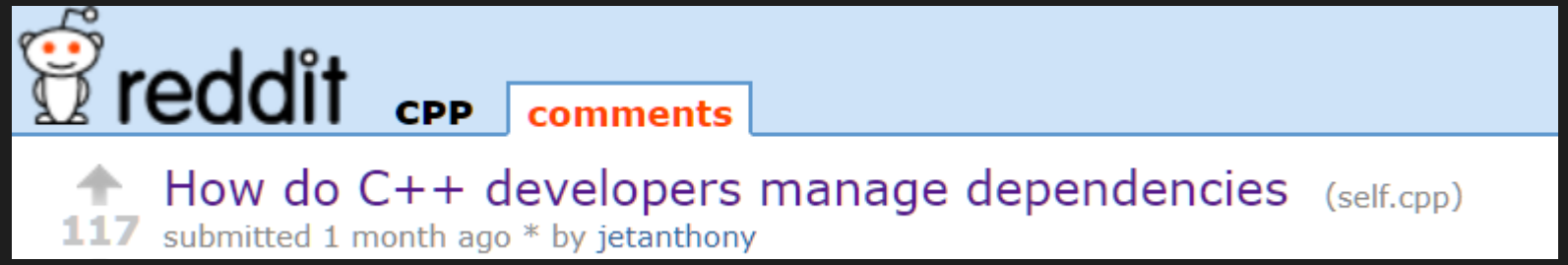
Its so painful in c++ that most devs i know rather reinvent any given functionality than import it from somewhere. I guess you could say c++ has built-in NIH support.

[–] **quzox_** 46 points 1 month ago

Step 1: Buy a lot of vodka

Step 2: Drink it

Step 3: Hope that the build is OK



[–] **ooglesworth** 151 points 1 month ago

Through much pain and anguish.

[–] **emthreeve** 7 points 1 month ago

From years of experience in several large companies: they don't, mostly.

[–] **Droggl** 22 points 1 month ago

Its so painful in c++ that most devs i know rather reinvent any given functionality than import it from somewhere. I guess you could say c++ has built-in NIH support.

[–] **Gustav__Mahler** 4 points 1 month ago

Multiple responses recommending hard liquor. My people.

[–] **quzox_** 46 points 1 month ago

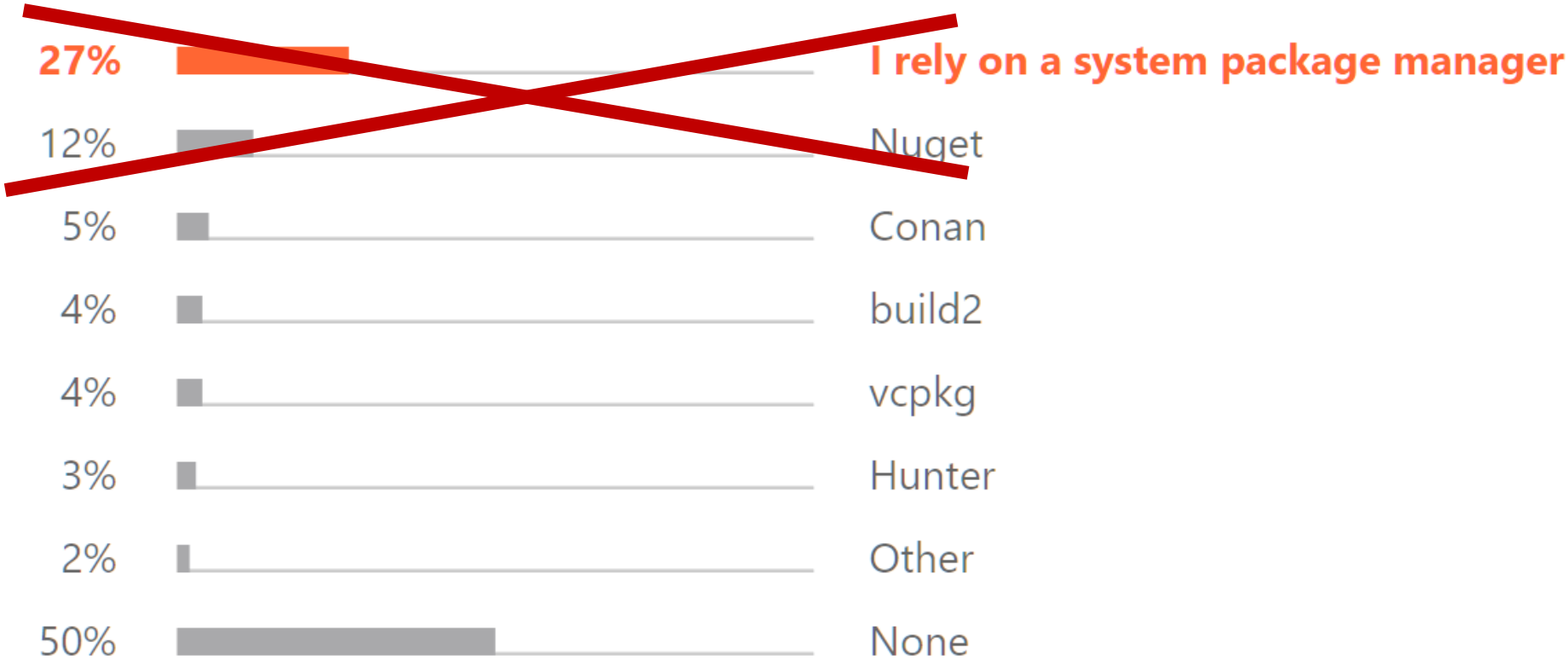
Step 1: Buy a lot of vodka

Step 2: Drink it

Step 3: Hope that the build is OK

What dependency managers do you use in your projects?

What dependency managers do you use in your projects?



Часть 1.

Что может быть проще vcpkg?

```
git clone https://github.com/Microsoft/vcpkg.git
cd vcpkg
bootstrap-vcpkg.bat
```

```
git clone https://github.com/Microsoft/vcpkg.git
cd vcpkg
bootstrap-vcpkg.bat

vcpkg.exe integrate install
```

```
git clone https://github.com/Microsoft/vcpkg.git
cd vcpkg
bootstrap-vcpkg.bat

vcpkg.exe integrate install
```

All MSBuild C++ projects can now #include any installed libraries.

Linking will be handled automatically.

Installing new libraries will make them instantly available.

```cpp
#include <fmt/format.h>

int main() {
    fmt::format("Hello {}", "vcpkg");
}
```

```cpp
#include <fmt/format.h>

int main() {
    fmt::format("Hello {}", "vcpkg");
}
```

```
vcpkg install fmt:x64-windows
```

\packages\fmt_x64-windows\

```
\packages\fmt_x64-windows\

include -> fmt\*.h
```

```
\packages\fmt_x64-windows\

include -> fmt\*.h
bin     -> fmt.dll, fmt.pdb
```

```
\packages\fmt_x64-windows\

include -> fmt\*.h
bin     -> fmt.dll, fmt.pdb
lib     -> fmt.lib
```

```
\packages\fmt_x64-windows\

include -> fmt\*.h
bin     -> fmt.dll, fmt.pdb
lib     -> fmt.lib
debug   -> lib, bin (fmtd.*)
```

```
\packages\fmt_x64-windows\

include -> fmt\*.h
bin     -> fmt.dll, fmt.pdb
lib     -> fmt.lib
debug   -> lib, bin (fmtd.*)
share   -> *.cmake, copyright\license
```

```
\packages\fmt_x64-windows\
    --Копирование-->
\installed\x64-windows\
```

```
\installed\x64-windows\

include -> fmt\*.h, boost\*.h
bin     -> fmt.dll, boost-*.dll
lib     -> fmt.lib, boost-*.lib
debug   -> lib, bin (fmtd.*, boost-*d.*)
share   -> *.cmake, copyrights\licenses
```

Часть 2.

Интеграция.

MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.Cpp.Common.props

```
<VCLibPackagePath Condition="'$(VCLibPackagePath)' == ''">
  $(LOCALAPPDATA)\vcpkg\vcpkg.user
</VCLibPackagePath>
```

MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.Cpp.Common.props

```
<VCLibPackagePath Condition="'$(VCLibPackagePath)' == ''">
  $(LOCALAPPDATA)\vcpkg\vcpkg.user
</VCLibPackagePath>
```

MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.CppCommon.targets

```
<Import
  Condition="'$(VCLibPackagePath)' != '' and Exists('$(VCLibPackagePath).targets')"
    Project="$(VCLibPackagePath).targets"
/>
```

```
MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.Cpp.Common.props

<VCLibPackagePath Condition="'$(VCLibPackagePath)' == ''">
  $(LOCALAPPDATA)\vcpkg\vcpkg.user
</VCLibPackagePath>

MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.CppCommon.targets

<Import
  Condition="'$(VCLibPackagePath)' != '' and Exists('$(VCLibPackagePath).targets')"
    Project="$(VCLibPackagePath).targets"
/>

%LOCALAPPDATA%\vcpkg\vcpkg.user.targets (C:\Users\%USERNAME%\Appdata\Local\...)

<Project>
  <Import Condition="Exists('D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets')"
     Project="D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets"
  />
</Project>
```

```
MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.Cpp.Common.props

<VCLibPackagePath Condition="'$(VCLibPackagePath)' == ''">
  $(LOCALAPPDATA)\vcpkg\vcpkg.user
</VCLibPackagePath>

MSVS_FOLDER/MSBuild/Microsoft/VC/v160/Microsoft.CppCommon.targets

<Import
  Condition="'$(VCLibPackagePath)' != '' and Exists('$(VCLibPackagePath).targets')"
    Project="$(VCLibPackagePath).targets"
/>

%LOCALAPPDATA%\vcpkg\vcpkg.user.targets (C:\Users\%USERNAME%\Appdata\Local\...)

<Project>
  <Import Condition="Exists('D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets')"
     Project="D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets"
  />
</Project>
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<PropertyGroup Condition="'$(Platform)|$(ApplicationType)' == 'x64||'">
  <VcpkgEnabled Condition="'$(VcpkgEnabled)' == ''">true</VcpkgEnabled>
  <VcpkgTriplet Condition="'$(VcpkgTriplet)' == ''">x64-windows</VcpkgTriplet>
</PropertyGroup>
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<PropertyGroup Condition="'$(Platform)|$(ApplicationType)' == 'x64||'">
  <VcpkgEnabled Condition="'$(VcpkgEnabled)' == ''">true</VcpkgEnabled>
  <VcpkgTriplet Condition="'$(VcpkgTriplet)' == ''">x64-windows</VcpkgTriplet>
</PropertyGroup>
```

test_project.vcxproj

```
<VcpkgEnabled>true</VcpkgEnabled>
<VcpkgTriplet>x64-windows-static</VcpkgTriplet>
<Import Project="D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets" />
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<VcpkgConfiguration
  Condition="'$(VcpkgConfiguration)' == ''">
    $(Configuration)
</VcpkgConfiguration>

<VcpkgNormalizedConfiguration
   Condition="$(VcpkgConfiguration.StartsWith('Debug'))">
     Debug
</VcpkgNormalizedConfiguration>
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<VcpkgConfiguration
  Condition="'$(VcpkgConfiguration)' == ''">
    $(Configuration)
</VcpkgConfiguration>

<VcpkgNormalizedConfiguration
   Condition="$(VcpkgConfiguration.StartsWith('Release'))">
     Release
</VcpkgNormalizedConfiguration>
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<VcpkgConfiguration
  Condition="'$(VcpkgConfiguration)' == ''">
    $(Configuration)
</VcpkgConfiguration>
```

test_project.vcxproj

```
<VcpkgConfiguration>Release</VcpkgConfiguration>
<Import Project="D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets" />
```

D:\vcpkg\scripts\buildsystems\msbuild\vcpkg.targets

```
<VcpkgRoot Condition="'$(VcpkgRoot)' == ''">
    $([MSBuild]::GetDirectoryNameOfFileAbove(
        $(MSBuildThisFileDirectory), .vcpkg-root)
    )\installed\$(VcpkgTriplet)\
</VcpkgRoot>
```

D:\vcpkg + \installed\$(VcpkgTriplet)\

D:\vcpkg\installed\x64-windows\

```
\installed\x64-windows\

include -> fmt\*.h, boost\*.h
bin     -> fmt.dll, boost-*.dll
lib     -> fmt\fmt.lib, boost-*.lib
debug   -> lib, bin (fmtd.*, boost-*d.*)
share   -> *.cmake, copyrights\licenses
```

```
<ClCompile>
  <AdditionalIncludeDirectories>
    %(AdditionalIncludeDirectories);$(VcpkgRoot)include
  </AdditionalIncludeDirectories>
</ClCompile>
```

```
<Link>
  <AdditionalDependencies>
    %(AdditionalDependencies);$(VcpkgRoot)lib\*.lib
  </AdditionalDependencies>
</Link>
```

```
VcpkgRoot: D:\vcpkg\installed\x64-windows\
Link: D:\vcpkg\installed\x64-windows\lib\*.lib
```

Автолинковка!

```
<Link>
  <AdditionalDependencies>
    %(AdditionalDependencies);$(VcpkgRoot)lib\*.lib
  </AdditionalDependencies>
</Link>
```

Страшный скрипт:

1. dumpbin $(TargetFile).$(TargetExt)
2. Ищем каждую библиотеку в $(VcpkgRoot)bin\
3. Копируем, при нахождении

Часть 3.

Версионирование библиотек?

%VCPKG_DIR%/ports/%PACKAGE%/CONTROL:

```
Source: imgui-sfml
Version: 2.1
Homepage: https://github.com/eliasdaler/imgui-sfml
Description: ImGui binding for use with SFML
Build-Depends: sfml, imgui
```

%VCPKG_DIR%/ports/%PACKAGE%/portfile.cmake:

```
include(vcpkg_common_functions)
vcpkg_from_git(
  OUT_SOURCE_PATH SOURCE_PATH
  URL https://gitlab-master.nvidia.com/dpanin/prebuilt_lib
  REF 383b691d47c921270c0a87887e2865fc2c571042)
```

%VCPKG_DIR%/ports/%PACKAGE%/portfile.cmake:

```
include(vcpkg_common_functions)
vcpkg_from_git(
  OUT_SOURCE_PATH SOURCE_PATH
  URL https://gitlab-master.nvidia.com/dpanin/prebuilt_lib
  REF 383b691d47c921270c0a87887e2865fc2c571042)

file(MAKE_DIRECTORY
  ${CURRENT_PACKAGES_DIR}/include
  ...)

file(COPY "${SOURCE_PATH}/include/prebuilt_lib.h"
  DESTINATION ${CURRENT_PACKAGES_DIR}/include/
) # Не забываем бинари
```

%VCPKG_DIR%/ports/%PACKAGE%/portfile.cmake:

```
include(vcpkg_common_functions)
vcpkg_from_git(
  OUT_SOURCE_PATH SOURCE_PATH
  URL https://gitlab-master.nvidia.com/dpanin/prebuilt_lib
  REF 383b691d47c921270c0a87887e2865fc2c571042)

file(MAKE_DIRECTORY
  ${CURRENT_PACKAGES_DIR}/include
  ...)

file(COPY "${SOURCE_PATH}/include/prebuilt_lib.h"
  DESTINATION ${CURRENT_PACKAGES_DIR}/include/
) # Не забываем бинари

file(WRITE ${CURRENT_PACKAGES_DIR}/share/prebuilt_lib/copyright
 "Copyright rofl")
```

```
somewhere_else/boost-1.34/{CONTROL, portfile.cmake}
somewhere_else/boost-1.35/{CONTROL, portfile.cmake}
```

```
somewhere_else/boost-1.34/{CONTROL, portfile.cmake}
somewhere_else/boost-1.35/{CONTROL, portfile.cmake}


vcpkg install boost-1.34 --overlay-ports=somewhere_else
```

```
somewhere_else/boost-1.34/{CONTROL, portfile.cmake}
somewhere_else/boost-1.35/{CONTROL, portfile.cmake}


vcpkg install boost-1.34 --overlay-ports=somewhere_else

vcpkg install boost-1.35 --overlay-ports=somewhere_else
```

```
somewhere_else/boost-1.34/{CONTROL, portfile.cmake}
somewhere_else/boost-1.35/{CONTROL, portfile.cmake}


vcpkg install boost-1.34 --overlay-ports=somewhere_else

vcpkg remove boost-1.34
vcpkg install boost-1.35 --overlay-ports=somewhere_else
```

```
somewhere_else/boost-1.34/{CONTROL, portfile.cmake}
somewhere_else/boost-1.35/{CONTROL, portfile.cmake}
```

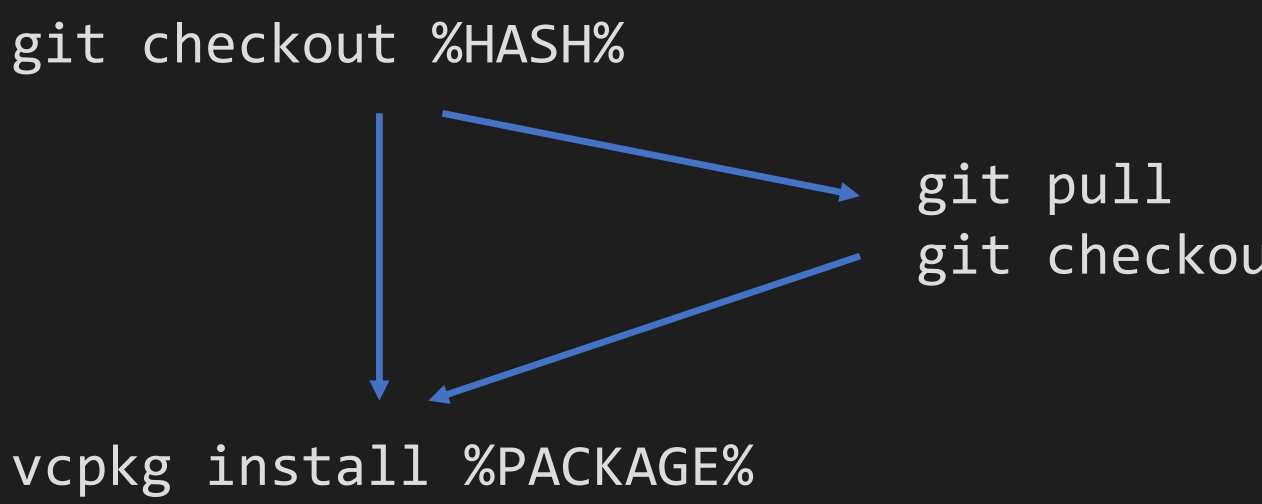
```
vcpkg install boost-1.34 --overlay-ports=somewhere_else
```

```
vcpkg remove boost-1.34
vcpkg install boost-1.35 --overlay-ports=somewhere_else
```

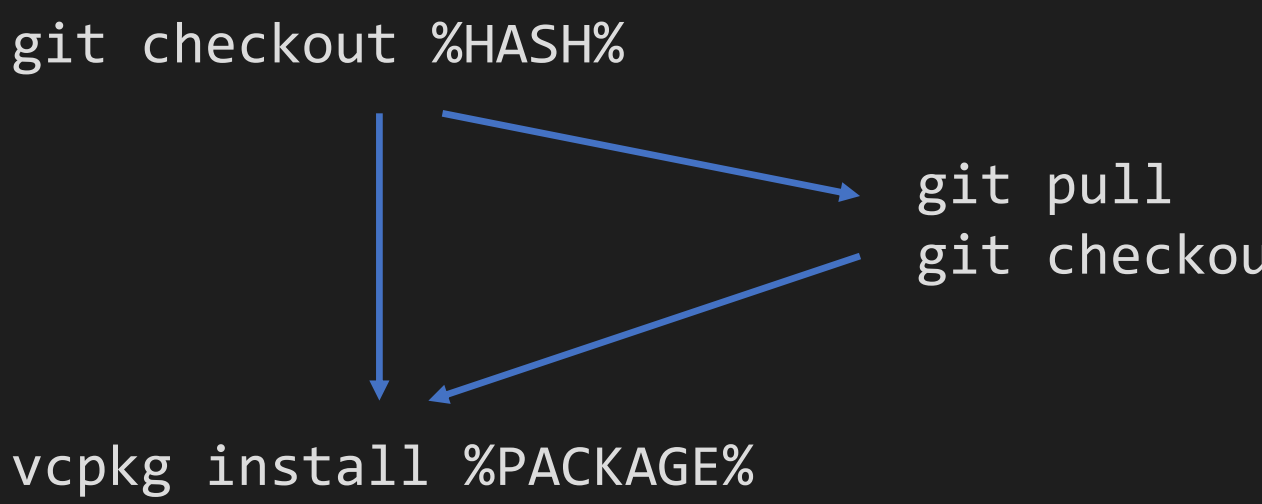
:(

```
somewhere_else/our-boost/{CONTROL, portfile.cmake}
```

somewhere_else/our-boost/{CONTROL, portfile.cmake}



somewhere_else/our-boost/{CONTROL, portfile.cmake}



somewhere_else/boost@HASH

```
vcpkg export packages... --raw --output=somewhere
```

```
vcpkg export packages... --raw --output=somewhere

somewhere\installed\%TRIPLET%\
    bin\
    include\
    lib\
    share\
```

```
vcpkg export packages... --raw --output=somewhere

somewhere\installed\%TRIPLET%\
    bin\
    include\
    lib\
    share\

somewhere\scripts\...
    vcpkg.targets
    vcpkg.cmake
```

```
vcpkg export packages... --raw --output=somewhere

<Project>
  <Import Project=
    "somewhere\...\vcpkg.targets"
</Project>
```

```
vcpkg export packages... --raw --output=somewhere
```

```
<Project>
  <Import Project=
    "somewhere\...\vcpkg.targets"
</Project>
```

```
Include    -> somewhere\...\include\
Link all   -> somewhere\...\lib\*.lib
Copy dll   -> somewhere\...\bin\*.dll
```

```
vcpkg_root = $(SolutionDir)\.nvpkg

vcpkg export packages... --raw --output=$(vcpkg_root)


Include   -> $(vcpkg_root)\...\include\
Link all  -> $(vcpkg_root)\...\lib\*.lib
Copy dll  -> $(vcpkg_root)\...\bin\*.dll
```

```
vcpkg_root = $(SolutionDir)\.nvpkg

vcpkg export packages... --raw --output=$(vcpkg_root)


Include   -> $(vcpkg_root)\...\include\
Link all  -> $(vcpkg_root)\...\lib\*.lib
Copy dll  -> $(vcpkg_root)\...\bin\*.dll
```

Не забываем почистить все за собой!
И как-то все заблокировать...

Каждому проекту свой vcpkg!

0. В .gitignore -> .vcpkg

Каждому проекту свой vcpkg!

0. В .gitignore -> .vcpkg

1. Prebuild - клон и сборка в $(SolutionDir)\.vcpkg

Каждому проекту свой vcpkg!

0. В .gitignore -> .vcpkg
1. Prebuild - клон и сборка в $(SolutionDir)\.vcpkg
2. Prebuild - чтение файла зависимостей?..

Часть 4.

Чтение файла зависимостей?

```
$(SolutionDir)\...\vcpkg.cpp

static const char* nvidia_vcpkg_packages = {
  /* REPO / Package @ hash */
  "fmt@..."
  "rapidjson@..."
  "nvpkg/internal@..."
};
```

```
$(SolutionDir)\...\vcpkg.cpp

static const char* nvidia_vcpkg_packages = {
  /* REPO / Package @ hash */
  "fmt@..."
  "rapidjson@..."
  "nvpkg/internal@..."
};
```

Изменение .cpp файла:

1. Вызов prebuild скрипта с подтягиванием зависимостей
2. перекомпиляция всех зависимых проектов

Pre-build скрипт

1. Проверка $(SolutionDir)\.vcpkg\vcpkg.exe
   Скачивание и сборка, если нету.

Pre-build скрипт

1. Проверка $(SolutionDir)\.vcpkg\vcpkg.exe
   Скачивание и сборка, если нету.

2. Проверка vcpkg.cpp внутри .vcpkg\installed
   Если идентичны - то уже все скачано

Pre-build скрипт

1. Проверка $(SolutionDir)\.vcpkg\vcpkg.exe
   Скачивание и сборка, если нету.

2. Проверка vcpkg.cpp внутри .vcpkg\installed
   Если идентичны - то уже все скачано

3. Удаляем installed, downloaded, packages

Pre-build скрипт

1. Проверка $(SolutionDir)\.vcpkg\vcpkg.exe
   Скачивание и сборка, если нету.

2. Проверка vcpkg.cpp внутри .vcpkg\installed
   Если идентичны - то уже все скачано

3. Удаляем installed, downloaded, packages

4. Комбо из git checkout %HASH% && vcpkg install %PACKAGE%

Что у нас в итоге получилось?

0. Не нужно ничего устанавливать

Что у нас в итоге получилось?

0. Не нужно ничего устанавливать

1. Скачивание & сборка vcpkg, установка пакетов в 1 клик RUN

Что у нас в итоге получилось?

0. Не нужно ничего устанавливать

1. Скачивание & сборка vcpkg, установка пакетов в 1 клик RUN

2. Перекачивание и пересборка автоматическая

Что у нас в итоге получилось?

0. Не нужно ничего устанавливать

1. Скачивание & сборка vcpkg, установка пакетов в 1 клик RUN

2. Перекачивание и пересборка автоматическая

3. Все детские болезни vcpkg успешно вылечены:
   - Версионирование работает
   - Одновременное наличие разных версий не нужно

Часть 5.

Бонус-контент.

```
project (hello)
include (~/vcpkg/scripts/buildsystems/vcpkg.cmake)

add_executable (hello hello.cpp)
find_package (fmt CONFIG REQUIRED)
target_link_libraries (hello PRIVATE fmt::fmt)
```

```
cmake . && cmake --build .
```

%VCPKG_ROOT%\triplets\nvidia-x64.cmake

```
set(VCPKG_TARGET_ARCHITECTURE x64)
set(VCPKG_CRT_LINKAGE dynamic)
set(VCPKG_LIBRARY_LINKAGE static)
```

%VCPKG_ROOT%\triplets\nvidia-x64.cmake

```
set(VCPKG_TARGET_ARCHITECTURE x64)
set(VCPKG_CRT_LINKAGE dynamic)
set(VCPKG_LIBRARY_LINKAGE static)

set(VCPKG_CMAKE_SYSTEM_NAME Linux)
set(VCPKG_CXX_FLAGS /Qspectre)
```

https://github.com/microsoft/vcpkg/blob/master/docs/users/triplets.md

Не хочу пересобирать каждый раз!

```
$(SolutionDir)\vcpkg.cpp
$(SolutionDir)\.vcpkg\installed\*
```

Не хочу пересобирать каждый раз!

```
$(SolutionDir)\vcpkg.cpp
$(SolutionDir)\.vcpkg\installed\*

$(SolutionName)-$(Arch)-$(Toolchain)-$(vcpkg_cpp_hash).zip -> Artifactory
```

Не хочу пересобирать каждый раз!

```
$(SolutionDir)\vcpkg.cpp
$(SolutionDir)\.vcpkg\installed\*

$(SolutionName)-$(Arch)-$(Toolchain)-$(vcpkg_cpp_hash).zip -> Artifactory


vcpkg install boost:boost-$(Arch)-$(Toolchain)-$(Hash)
.vcpkg\packages\boost_boost-$(Arch)-$(Toolchain)-$(Hash) \*

$(Package)-$(Arch)-$(Toolchain)-$(Hash).zip -> Artifactory
```

Не хочу пересобирать каждый раз!

```
$(SolutionDir)\vcpkg.cpp
$(SolutionDir)\.vcpkg\installed\*

$(SolutionName)-$(Arch)-$(Toolchain)-$(vcpkg_cpp_hash).zip -> Artifactory


vcpkg install boost:boost-$(Arch)-$(Toolchain)-$(Hash)
.vcpkg\packages\boost_boost-$(Arch)-$(Toolchain)-$(Hash) \*

$(Package)-$(Arch)-$(Toolchain)-$(Hash).zip -> Artifactory


boost-x64-v142-HASH.zip     ->
fmt-x64-v142-HASH2.zip      ->    .vcpkg\installed\nvidia-x64\
sfml-x64-v142-HASH3.zip     ->
```

Часть 6.

Которую все ждали.

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)

2. Простая автоматизация внутренних релизов

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)
2. Простая автоматизация внутренних релизов
3. Восхитительная интеграция с Visual Studio

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)
2. Простая автоматизация внутренних релизов
3. Восхитительная интеграция с Visual Studio
4. Conan не умеет в автоперекачку зависимостей

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)
2. Простая автоматизация внутренних релизов
3. Восхитительная интеграция с Visual Studio
4. Conan не умеет в автоперекачку зависимостей
5. Интеграция Conan'а меняет .vcxproj файлы

Часть 6, которую вы все ждали.

1. Количество пакетов (1225+ vs 300+)
2. Простая автоматизация внутренних релизов
3. Восхитительная интеграция с Visual Studio
4. Conan не умеет в автоперекачку зависимостей
5. Интеграция Conan'а меняет .vcxproj файлы
6. Я еще не доучил C++, чтобы учить Python

vcpkg install the_end:2019-piter

github.com/starl1ght/piter2019

t.me/starl1ght