Sharing Toolchain Files Across Beman Projects

How Beman.exemplar uses CMake Toolchain files

exemplar / cmake /

bretbrownjr Clean up install logic

Name
..
appleclang-toolchain.cmake
gnu-toolchain.cmake
llvm-toolchain.cmake
msvc-toolchain.cmake
use-fetch-content.cmake

```
include_guard(GLOBAL)

set(CMAKE_C_COMPILER gcc)
set(CMAKE_CXX_COMPILER g++)

if(BEMAN_BUILDSYS_SANITIZER STREQUAL "MaxSan")
    set(SANITIZER_FLAGS
        "-fsanitize=address -fsanitize=leak -fsanitize=pointer-compare -fsanitize=pointer-subtract -fsanitize=undefined -fsanitize-undefined-trap-on-error"
    )
elseif(BEMAN_BUILDSYS_SANITIZER STREQUAL "TSan")
    set(SANITIZER_FLAGS "-fsanitize=thread")
endif()

set(CMAKE_C_FLAGS_DEBUG_INIT "${SANITIZER_FLAGS}")
set(CMAKE_CXX_FLAGS_DEBUG_INIT "${SANITIZER_FLAGS}")

set(RELEASE_FLAGS "-O3 ${SANITIZER_FLAGS}")

set(CMAKE_C_FLAGS_RELWITHDEBINFO_INIT "${RELEASE_FLAGS}")
set(CMAKE_CXX_FLAGS_RELWITHDEBINFO_INIT "${RELEASE_FLAGS}")

set(CMAKE_C_FLAGS_RELEASE_INIT "${RELEASE_FLAGS}")
set(CMAKE_CXX_FLAGS_RELEASE_INIT "${RELEASE_FLAGS}")
```

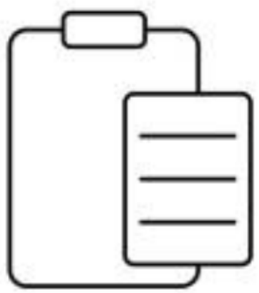
```
{
  "name": "gcc-debug",
  "displayName": "GCC Debug Build",
  "inherits": [
    "_root-config",
    "_debug-base"
  ],
  "cacheVariables": {
    "CMAKE_TOOLCHAIN_FILE": "cmake/gnu-toolchain.cmake"
  }
},
```

Status Quo

exemplar / cmake /

bretbrownjr Clean up install logic

Name
..
appleclang-toolchain.cmake
gnu-toolchain.cmake
llvm-toolchain.cmake
msvc-toolchain.cmake
use-fetch-content.cmake



18 repositories

task Public
Implementation of the coroutine task lazy26 (P3552)
C++ · Other · 3 · 3 · 1 · 1 · Updated 1 hour ago

infra Public
Internal Beman Project infrastructure repo
Dockerfile · 6 · 2 · 4 · 2 · Updated 1 hour ago

utf_view Public
Implementation of the UTF transcoding views proposal, P2728
C++ · Boost Software License 1.0 · 4 · 8 · 2 · 0 · Updated 13 hours ago

any_view Public
A generalized type-erased view with customizable properties (P3411)
C++ · Other · 3 · 4 · 4 · 1 · Updated yesterday

execution Public
Implementation of the std::execution (P2300, sender/receiver) proposal
C++ · Other · 14 · 68 · 19 · 6 · Updated 2 days ago

exemplar Public template
A Beman Library Exemplar (Template)
CMake · Other · 25 · 44 · 13 · 14 · Updated 2 days ago

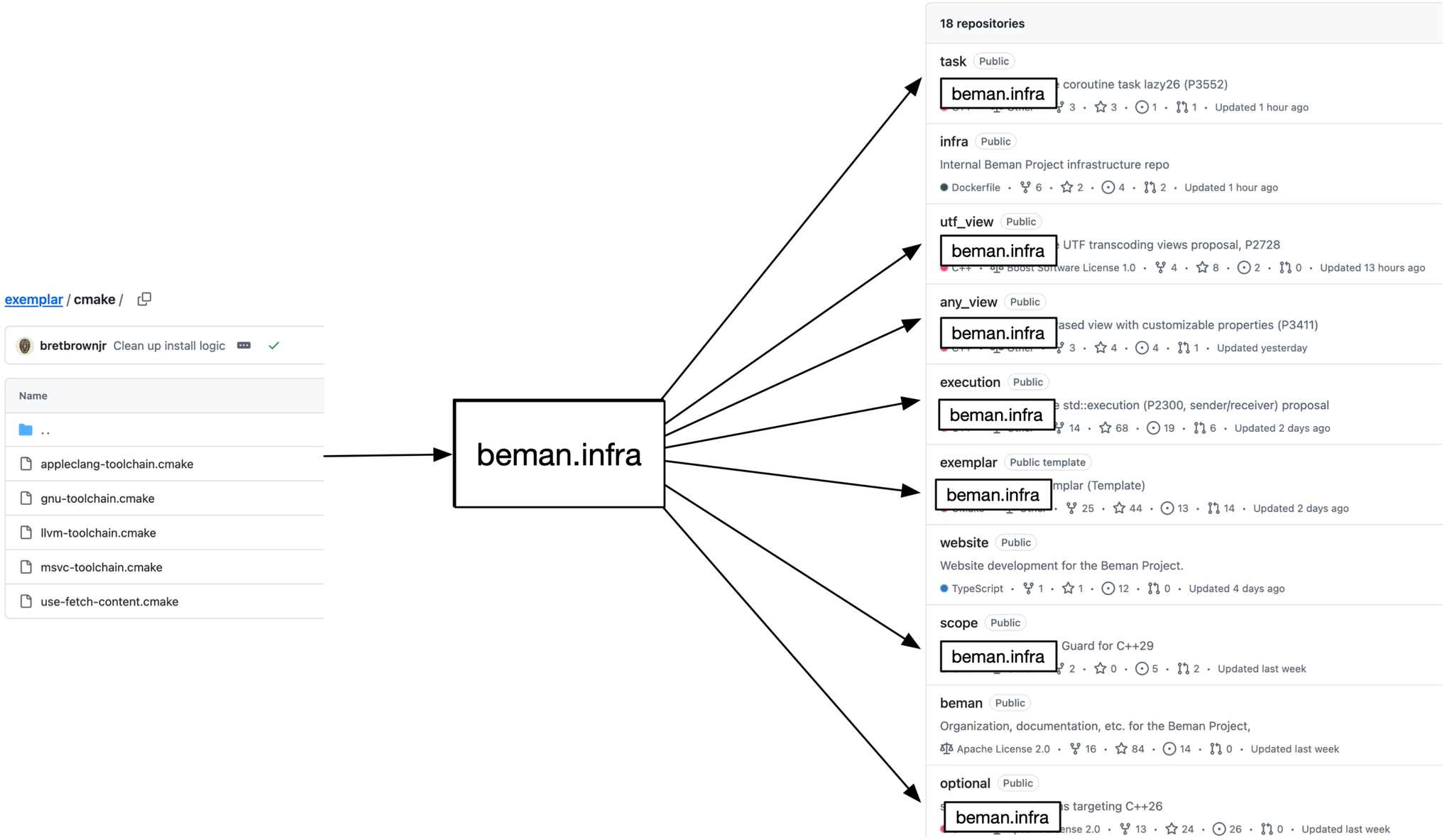
website Public
Website development for the Beman Project.
TypeScript · 1 · 1 · 12 · 0 · Updated 4 days ago

scope Public
P3610 Generic Scope Guard for C++29
C++ · Other · 2 · 0 · 5 · 2 · Updated last week

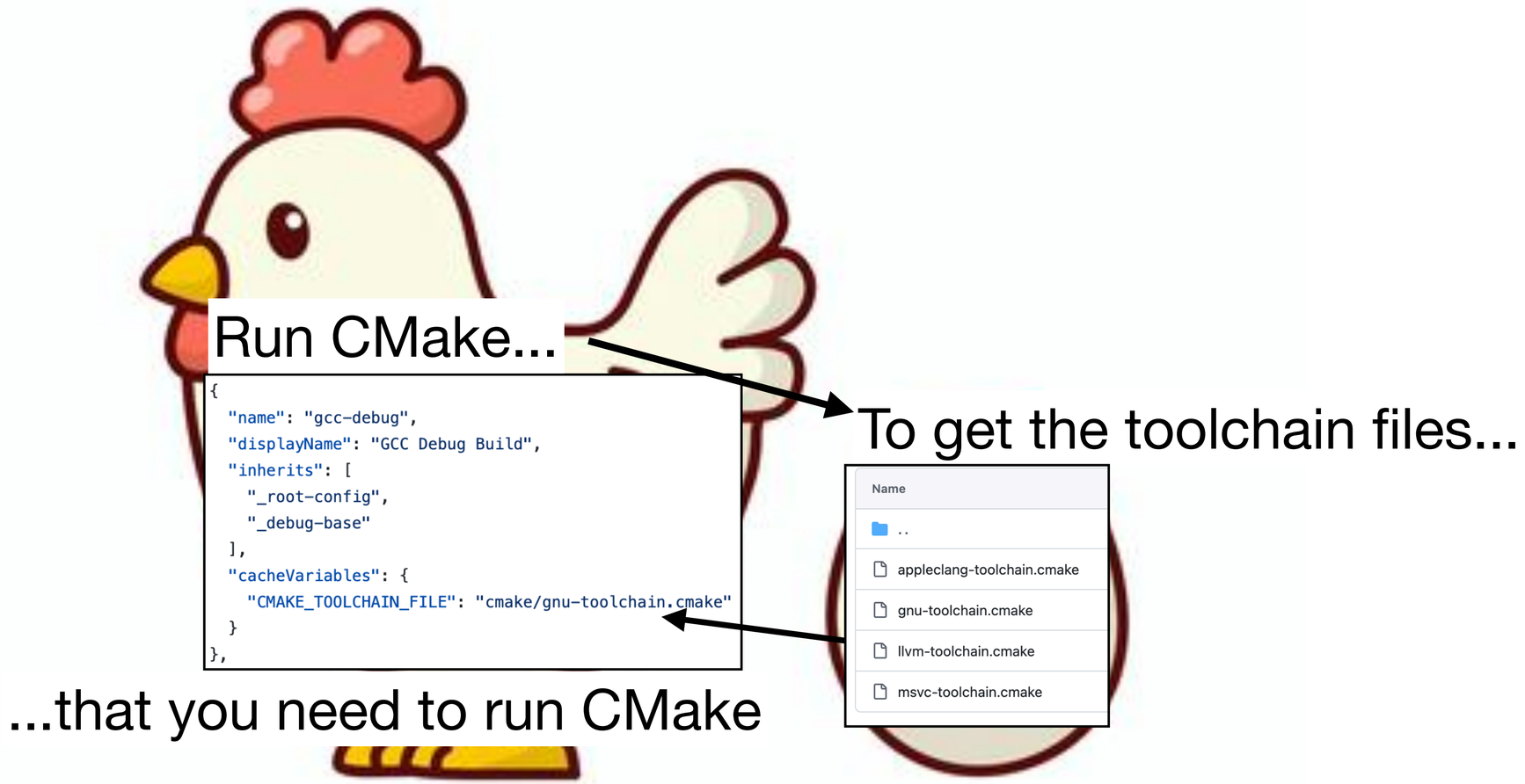
beman Public
Organization, documentation, etc. for the Beman Project,
Apache License 2.0 · 16 · 84 · 14 · 0 · Updated last week

optional Public
std::optional extensions targeting C++26
C++ · Apache License 2.0 · 13 · 24 · 26 · 0 · Updated last week

Move the toolchain files into beman.infra



CMake can't give us this dependency



Option 1: Git Subtree

On the other hand, git subtrees take advantage of git's support for merging together unrelated commit histories in order to incorporate the git commits of the dependency into the commit graph of the parent. For example, imagine we have a git repo called `parent` with the following history:

```
o Frobnicate widgets
|
o Reticulate splines
|
o Initial commit
```

And a repo called `dependency` with the history:

```
o Reinitialize enigmas
|
o Calibrate flux capacitors
|
o Initial commit
```

Git Subtree Continued

When `dependency` is added as a subtree, the repository's history will look like:

```
o Add 'dependency/' from commit '12345abcdef'
|\
| \
|  o Reinitialize enigmas
|  |
|  o Calibrate flux capacitors
|  |
|  o Initial commit
|
o Frobnicate widgets
|
o Reticulate splines
|
o Initial commit
```

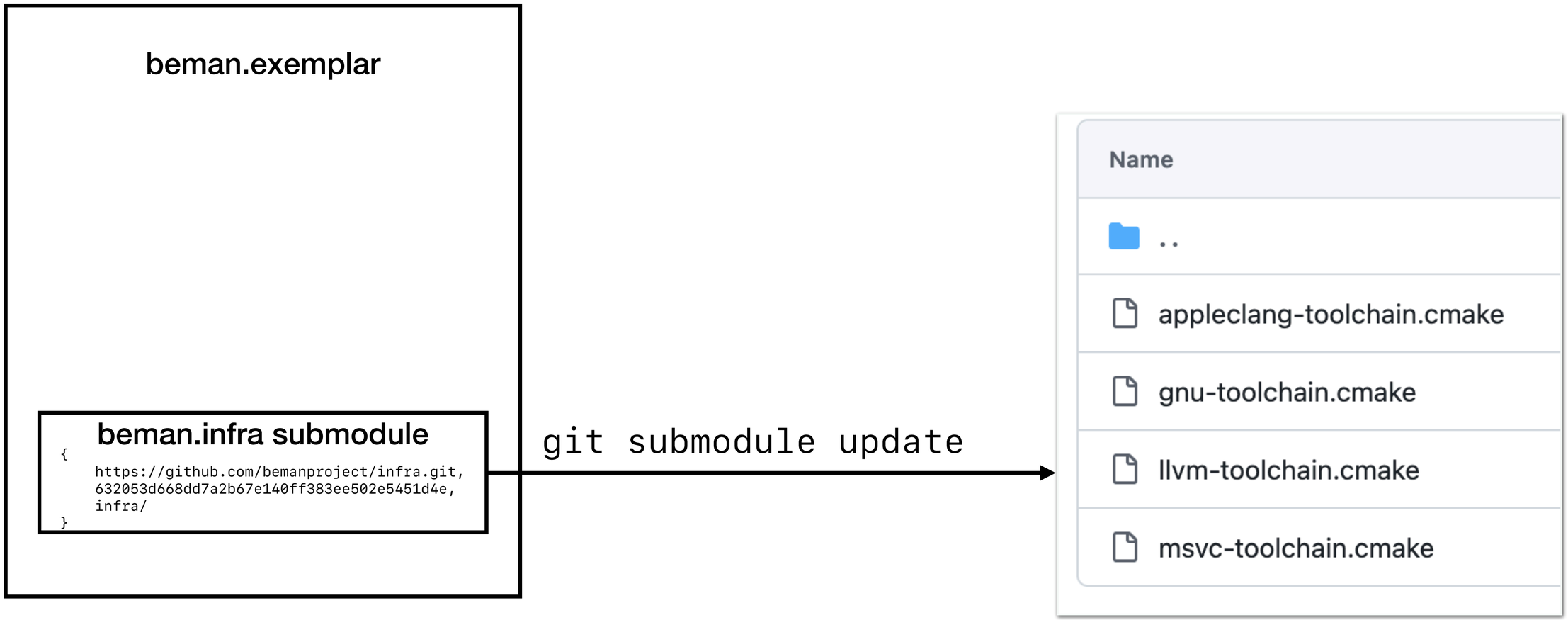
When a further change is made to `dependency` that needs to be brought in to `parent`, it's incorporated as a merge commit:

```
o Merge commit '54321fedcba'
|\
| \
|  o Ameliorate checksums
|  |
o  | Add 'dependency/' from commit '12345abcdef'
|\ |
| \|
|  o Reinitialize enigmas
|  |
|  o Calibrate flux capacitors
|  |
|  o Initial commit
|
o Frobnicate widgets
|
o Reticulate splines
|
o Initial commit
```

Problems with git subtree

- No single source of truth
- Requires merge commits
- Messy git history
- Doesn't work like a normal dependency

Option 2: git submodule



Advantages of git submodules

- No messy git history full of merge commits
- Single source of truth
- Works like other dependencies

Objection to git submodules #1: User Experience

```
~/sync/cpp/bemanproject/exemplar →enolan_infrasubmodule1* $ cmake --workflow --preset gcc-debug
Executing workflow step 1 of 3: configure preset "gcc-debug"

Preset CMake variables:

  BEMAN_BUILDSYS_SANITIZER="MaxSan"
  CMAKE_BUILD_TYPE="Debug"
  CMAKE_CXX_STANDARD="20"
  CMAKE_EXPORT_COMPILE_COMMANDS="ON"
  CMAKE_PROJECT_TOP_LEVEL_INCLUDES="./cmake/use-fetch-content.cmake"
  CMAKE_TOOLCHAIN_FILE="infra/cmake/gnu-toolchain.cmake"

CMake Error at /home/eddie/localbuild/cmake/share/cmake-3.31/Modules/CMakeDetermineSystem.cmake:152 (message):
  Could not find toolchain file: infra/cmake/gnu-toolchain.cmake
Call Stack (most recent call first):
  CMakeLists.txt:5 (project)


CMake Error: CMake was unable to find a build program corresponding to "Ninja".  CMAKE_MAKE_PROGRAM is not set.  You probably need to select
 a different build tool.
CMake Error: CMAKE_CXX_COMPILER not set, after EnableLanguage
-- Configuring incomplete, errors occurred!
```

You should have run `git submodule update --init` or `git clone --recursive`

Objection to git submodules #2: GitHub releases

Presets to FetchContent

Latest

bretbrownjr released this 3 days ago 2.1.1 0a275de

About

This change sets all CMake presets to use FetchContent explicitly. Note that t
Please use explicit, step-by-step CMake workflows to build in situations wher

Release Maturity

This release has a status of "under development". It is not recommended for
maturity status implies.

What's Changed

- Set CMAKE_PROJECT_TOP_LEVEL_INCLUDES in base preset by @steve

Full Changelog: 2.1.0...2.1.1

Contributors

steve-downey

Assets 2

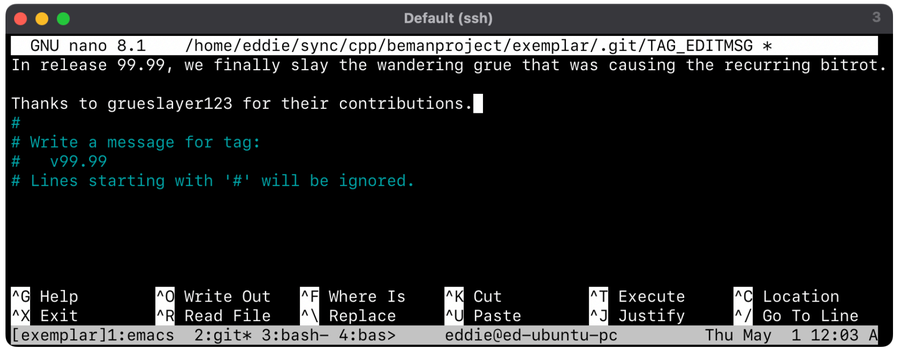
Source code (zip)

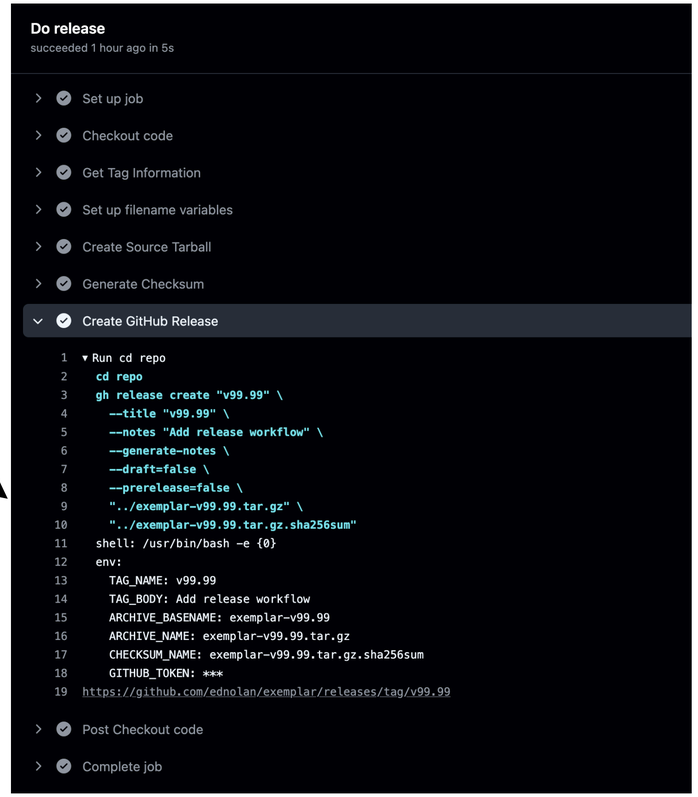
Source code (tar.gz)

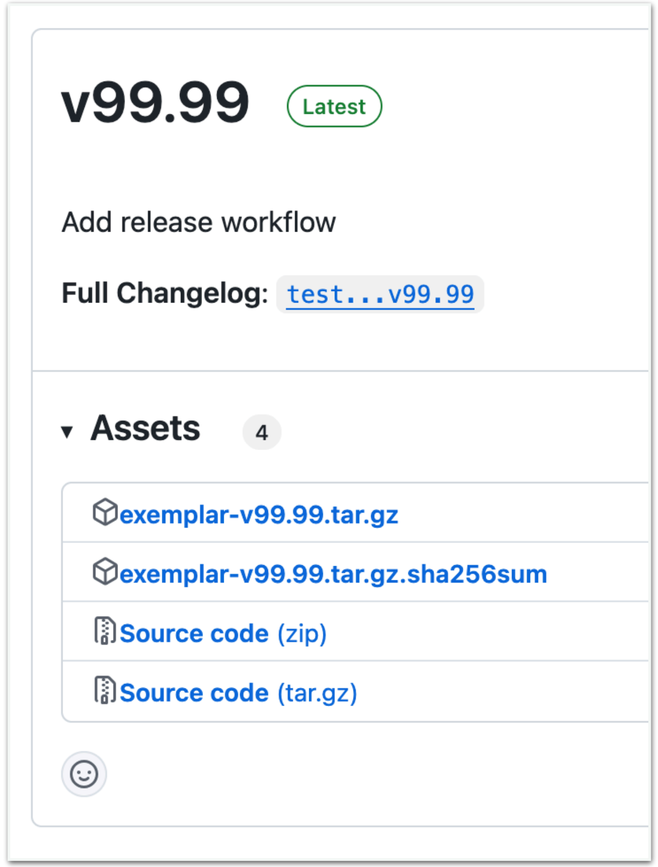
```
/tmp/exemplar $ tree .
.
├── cmake
│   └── use-fetch-content.cmake
├── CMakeLists.txt
├── CMakePresets.json
├── docs
│   ├── README.md
│   └── using_exemplar.md
├── examples
│   ├── CMakeLists.txt
│   ├── identity_as_default_projection.cpp
│   └── identity_direct_usage.cpp
├── include
│   └── beman
│       └── exemplar
│           └── identity.hpp
├── infra
├── LICENSE
├── lockfile.json
├── README.md
├── src
│   └── beman
│       └── exemplar
│           ├── beman.exemplar-config.cmake.in
│           ├── CMakeLists.txt
│           └── identity.cpp
└── tests
    └── beman
        └── exemplar
            ├── CMakeLists.txt
            └── identity.test.cpp

14 directories, 17 files
```

Objection to git submodules #2: GitHub releases

```
~/sync/cpp/bemanproject/exemplar →enolan_releaseaction1* $ git tag -a v99.99
~/sync/cpp/bemanproject/exemplar →enolan_releaseaction1* $ git push ednolan tag v99.99
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 249 bytes | 249.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:ednolan/exemplar.git
 * [new tag]         v99.99 -> v99.99
~/sync/cpp/bemanproject/exemplar →enolan_releaseaction1* $
```







Objection to git submodules #2: GitHub releases

2 hours ago

github-actions

v99.99

3936d93

Compare

v99.99 Latest

Add release workflow

Full Changelog: `test...v99.99`

Assets 4

exemplar-v99.99.tar.gz	2 hours ago
exemplar-v99.99.tar.gz.sha256sum	2 hours ago
Source code (zip)	2 hours ago
Source code (tar.gz)	2 hours ago

```
/tmp/exemplar $ tree .
.
├── cmake
│   └── use-fetch-content.cmake
├── CMakeLists.txt
├── CMakePresets.json
├── docs
│   ├── README.md
│   └── using_exemplar.md
├── examples
│   ├── CMakeLists.txt
│   ├── identity_as_default_projection.cpp
│   └── identity_direct_usage.cpp
├── include
│   └── beman
│       └── exemplar
│           └── identity.hpp
├── infra
├── LICENSE
├── lockfile.json
├── README.md
├── src
│   └── beman
│       └── exemplar
│           ├── beman.exemplar-config.cmake.in
│           ├── CMakeLists.txt
│           └── identity.cpp
└── tests
    └── beman
        └── exemplar
            ├── CMakeLists.txt
            └── identity.test.cpp

14 directories, 17 files
```

Objection to git submodules #2: GitHub releases

No way to disable auto-generated sc #6003

Answered by MylesBorins **kornelski** asked this question in Genera

kornelski on Oct 5, 2021 edited

GitHub automatically adds "Source Code.zip" and "Source Code.tar.gz" li every release. These are problematic for a few reasons:

1. When I make releases for non-technical audience (a binary build of my software), these links are unwanted. They add clutter and confuse nor developer users.
2. When my repo uses git submodules, these archives lack submodules' code, which causes confusing compilation errors. I can't even *replace* my own archives that contain complete source code.
3. The source tarballs are not even suitable for distribution of many proje source code form, because they don't support inclusion of auto-gene (e.g. `./configure` or compiled data files). Auto-generated files shoul under version control — that causes repo bloat and merge pains.

So from my perspective the Source Code links are unnecessary, often unc and even when I'd like some source code links, the auto-generated archiv incomplete and unfit for the purpose.

Please remove these links or at least add an option to disable them.

↑ 107 67 1 1 2

v1.0.27: libusb 1.0.27

Compare

tormodvolden released this Jan 31, 2024 · 64 commits to master since this release v1.0.27

d52e355

Note: The Windows binaries in libusb-1.0.27.7z are provided as-is and may not work for your toolchain. In that case, please build libusb from source using your own toolchain.

Note:
*.tar.bz are our official release files, these are proper release tarballs including configure.

*.tar.gz and .zip are something GitHub generates, unfortunately we haven't found a way to disable them. They are just a git snapshot, without e.g. configure.

*) https://github.com/orgs/community/discussions/6003

Assets 8

libusb-1.0.27.7z	3.91 MB	Jan 31, 2024
libusb-1.0.27.7z.asc	833 Bytes	Jan 31, 2024
libusb-1.0.27.tar.bz2	629 KB	Jan 31, 2024
libusb-1.0.27.tar.bz2.asc	833 Bytes	Jan 31, 2024
SIGNING_KEYS.txt	905 Bytes	Jul 29, 2024
SIGNING_KEYS.txt.asc	833 Bytes	Jul 29, 2024
Source code (zip)		Jan 31, 2024
Source code (tar.gz)		Jan 31, 2024

22 6 7 1 2 30 people reacted

lked about internally. We would need to
stems that rely on these artifacts that
fore we could follow through on this.

are looking at a number of
ts / assets, but we may not be funding
pectations)

2 13 replies

Option 3: Reinventing the wheel

Check bemanmodules

failed 3 hours ago in 4s

- Set up job
- Checkout
- Run bemanmodule check

```
1  ▸ Run ./infra/bemanmodule.py --check
4  Mismatch between /home/runner/work/exemplar/exemplar/infra and https://github.com/bemanproject/infra.git
5  Found bemanmodules at:  ['/home/runner/work/exemplar/exemplar/infra']
6  Checking /home/runner/work/exemplar/exemplar/infra equivalence with https://github.com/bemanproject/infra
7  Error: Process completed with exit code 1.
```

beman.infra bemanmodule

```
bemanproject/exemplar $ cat infra/.bemanmodule
https://github.com/bemanproject/infra.git
2164f14d1f8e35644f14678f5ee264155f57209b
bemanproject/exemplar $
```

bemanmodule.py

--pull

--check

```python
#!/usr/bin/env python3

import argparse
import filecmp
import os
import pathlib
import shutil
import subprocess
import sys
import tempfile

def directory_compare(dir1, dir2, ignore):
    compared = filecmp.dircmp(dir1, dir2, ignore=ignore)
    if compared.left_only or compared.right_only or compared.diff_files:
        return False
    for common_dir in compared.common_dirs:
        path1 = os.path.join(dir1, common_dir)
        path2 = os.path.join(dir2, common_dir)
        if not directory_compare(path1, path2, ignore):
            return False
    return True

class Submodule:
    def __init__(self, dirpath, url, commit_hash):
        self.dirpath = dirpath
        self.url = url
        self.commit_hash = commit_hash

def get_submodule(dir):
    with open(os.path.join(dir, ".bemanmodule"), 'r') as f:
        return Submodule(dir, f.readline().strip(), f.readline().strip())

def find_submodule_dirs_in(dir):
    assert os.path.isdir(dir)
    result = []
    for dirpath, _, filenames in os.walk(dir):
        if ".bemanmodule" in filenames:
            result.append(dirpath)
    return result

def pull_submodule_into_tmpdir(submodule):
    tmpdir = tempfile.TemporaryDirectory()
    subprocess.run(
        ["git", "clone", submodule.url, tmpdir.name], capture_output=True, check=True)
    subprocess.run(
        ["git", "-C", tmpdir.name, "reset", "--hard", submodule.commit_hash],
        capture_output=True, check=True)
    return tmpdir

def submodule_pull(submodule):
    print(
        "Pulling", submodule.url, "at commit", submodule.commit_hash, "to",
        submodule.dirpath)
    tmpdir = pull_submodule_into_tmpdir(submodule)
    shutil.rmtree(os.path.join(tmpdir.name, ".git"))
    shutil.copytree(tmpdir.name, submodule.dirpath, dirs_exist_ok=True)

def submodule_check(submodule):
    print(
        "Checking", submodule.dirpath, "equivalence with", submodule.url, "at commit",
        submodule.commit_hash)
    tmpdir = pull_submodule_into_tmpdir(submodule)
    if not directory_compare(tmpdir.name, submodule.dirpath, [".bemanmodule", ".git"]):
        print(
            "Mismatch between", submodule.dirpath, "and", submodule.url, "at commit",
            submodule.commit_hash, file=sys.stderr)
        sys.exit(1)

def main():
    parser = argparse.ArgumentParser(description="Beman pseudo-submodule tool")
    parser.add_argument('--pull', help="Update all pseudo-submodules to latest main",
                        action="store_true")
    parser.add_argument('--check', help="Check pseudo-submodule consistency",
                        action="store_true")
    args = parser.parse_args();
    script_path = pathlib.Path(__file__).resolve().parent
    script_parent = script_path.parent
    submodule_dirs = find_submodule_dirs_in(script_parent)
    print("Found bemanmodules at: ", submodule_dirs)
    submodules = [get_submodule(dir) for dir in submodule_dirs]
    for submodule in submodules:
        if args.pull:
            submodule_pull(submodule)
        elif args.check:
            submodule_check(submodule)
        else:
            raise Exception("Specify either --pull or --check")

if __name__ == "__main__":
    main()
```

Option 3: Reinventing the wheel

- Pros:
  - beman.infra files are always present; no file-not-found errors *
  - GitHub autogenerated source tarballs aren't broken *
- Cons:
  - It's a hack
  - Maintenance/training burden of bemanmodule.py
  - No equivalent of `git submodule add` (need to manually copy over bemanmodule.py to get the infra module working)

*git subtrees also solve these, but I still don't like them

Option 4: Let bx handle it

```
./bx cmake -B build -S . -DCMAKE_CXX_STANDARD=20
```