Published in The Startup You have 1 free member-only story left this month. Sign up for Medium and get an extra one Leonardo Lima Aug 10, 2020 · 3 min read · 

Member-only · 

Listen CPM: An Awesome Dependency Manager for C++ with CMake  $\bigcirc$ C++ doesn't have a default package manager, making the development with external libraries complex, especially if you're making a multiplatform system. Because of this absence, the community has created some strategies to work around and there are even some pretty consolidated package managers out there, such as <u>Conan</u> and <u>Vcpkg</u>, but none of them are simple and easy to use like <u>CPM</u> (no, we aren't talking about <u>this one</u>). CPM is the new kid on the block, its first version was released on April 2019, it's a CMake script that gives CMake the capacity to make version control, caching, and install the dependencies of your project, created by <u>Lars</u> Melchior. Usually, on a modern C++ and CMake project, the libraries are installed on the system and find\_package() is called to load the library configuration, forcing every developer working on the project to have the same environment (sometimes with different tools' versions, giving us a nasty headache). CPM has a simple syntax and it's pretty straightforward, making our lives as devs way easier. We call CPMAddPackage to install an external library (or CPMFindPackage to search the system, and, if it's not found, install it). There are two ways to download a library: through a Github repository or a link. Let's use <u>Nlohmann's JSON</u> library as an example: 1 CPMAddPackage( NAME nlohmann\_json GITHUB\_REPOSITORY nlohmann/json **VERSION** 3.6.1) **CPMAddPackage.cmake** hosted with ♥ by **GitHub** view raw Or through a link: 1 CPMAddPackage( NAME nlohmann\_json **VERSION 3.6.1** URL https://github.com/nlohmann/json/releases/download/v3.6.1/include.zip URL HASH SHA256=69cc88207ce91347ea530b227ff0776db82dcb8de6704e1a3d74f4841bc651cf) 7 if(nlohmann\_json\_ADDED) add\_library(nlohmann\_json INTERFACE) target\_include\_directories(nlohmann\_json INTERFACE \${nlohmann\_json\_SOURCE\_DIR}) 10 endif() **CPMAddPackageLink.cmake** hosted with ♥ by **GitHub** view raw (You will need to use add\_library() and target\_include\_directories() if you aren't using the Github feature). All right, to understand the simplicity and ease of CPM let's create a small project and do a quick example using spdlog, made by Gabi Melman. This is how the source tree will look like: src |\_\_\_main.cpp CMakeLists.txt This is our CMakeLists.txt: 1 cmake\_minimum\_required(VERSION 3.18) 3 project(cpm\_example) 5 set(CMAKE\_CXX\_STANDARD 17) 7 add\_executable(cpm\_example src/main.cpp) **CMakeMinumum.cmake** hosted with ♥ by **GitHub** First, we have to download CPM's script on our project and include it on CMake. There are two ways to do this. The first one is manually downloading it with the following commands: \$ mkdir -p cmake \$ wget -0 cmake/CPM.cmake https://github.com/TheLartians/CPM.cmake/releases/latest/download/CPM. Then adding the line include(cmake/CPM.cmake) on CMakeLists.txt. The second way is adding the following snippet to CMake, which will check if CPM exists everytime the project builds (and download it if it doesn't exist): 1 set(CPM\_DOWNLOAD\_VERSION 0.27.2) 2 set(CPM\_DOWNLOAD\_LOCATION "\${CMAKE\_BINARY\_DIR}/cmake/CPM\_\${CPM\_DOWNLOAD\_VERSION}.cmake") 4 if(NOT (EXISTS \${CPM\_DOWNLOAD\_LOCATION})) message(STATUS "Downloading CPM.cmake") file(DOWNLOAD https://github.com/TheLartians/CPM.cmake/releases/download/v\${CPM\_DOWNLOA 7 endif() include(\${CPM\_DOWNLOAD\_LOCATION}) **InstallCPM.cmake** hosted with ♥ by **GitHub** view raw Then, our source tree will look like this: build \_\_cmake CPM.cmake src \_main.cpp CmakeListst.txt To implement spdlog to our project, just add the following snippet to CMakeLists.txt, after including CPM: 1 CPMAddPackage( NAME spdlog GITHUB\_REPOSITORY gabime/spdlog **VERSION 1.7.0) CPMAddPackageSpdlog.cmake** hosted with ♥ by **GitHub** view raw Then we have to link the library to the project: 1 target\_link\_libraries(cpm\_example **TargetLinkLibrariesSpdlog.cmake** hosted with ♥ by **GitHub** In the end, our CMakeLists.txt should look like this: 1 cmake\_minimum\_required(VERSION 3.18) project(cpm\_example) 5 set(CMAKE\_CXX\_STANDARD 17) 7 set(CPM\_DOWNLOAD\_VERSION 0.27.2) 8 set(CPM\_DOWNLOAD\_LOCATION "\${CMAKE\_BINARY\_DIR}/cmake/CPM\_\${CPM\_DOWNLOAD\_VERSION}.cmake") 10 if(NOT (EXISTS \${CPM\_DOWNLOAD\_LOCATION})) message(STATUS "Downloading CPM.cmake") file(DOWNLOAD https://github.com/TheLartians/CPM.cmake/releases/download/v\${CPM\_DOWNLO 13 endif() include(\${CPM\_DOWNLOAD\_LOCATION}) CPMAddPackage( NAME spdlog GITHUB\_REPOSITORY gabime/spdlog 20 **VERSION 1.7.0)** 21 22 add\_executable(cpm\_example src/main.cpp) 23 24 target\_link\_libraries(cpm\_example spdlog) CMakeCPMExample.cmake hosted with ♥ by GitHub view raw Now in our main, let's use spdlog to output a "Hello World!": 1 #include <spdlog/spdlog.h> 3 int main() { spdlog::info("Hello world!"); return 0; **SpdlogExample.cpp** hosted with ♥ by **GitHub** view raw And that's it, now we have to build the project and CPM will do all the hard work of downloading and showing CMake the path of our dependencies. So let's create a build directory, build the project, and run it: \$ mkdir -p build \$ cd build \$ cmake .. \$ make -j4 \$ ./cpm\_example Since CPM is a dependency manager and not a package manager, if you're working on multiple projects that use the same dependencies, you can easily set up different versions of the same library. The only downside of CPM is that every time you run a git clean -fdx or delete the build folder, it will remove the dependencies directory, and you will have to download everything again the next time you run cmake. But CPM lets you use a cache directory, passing the parameter -DCPM\_SOURCE\_CACHE to CMake and specifying a path. Like this: \$ cmake -DCPM\_SOURCE\_CACHE=/tmp/deps .. With just a few lines of CMake code, we can manage our external dependencies of a project, making environment configuration so much easier. CPM gives you a great advantage if you are working with a multiplatform project, CI tools (such as <u>Travis</u> and <u>Appveyor</u>), building inside a container, or any other situation where you have to set up a new environment from zero. 108 Sign up for Top 5 Stories By The Startup Get smarter at building your thing. Join 176,621+ others who receive The Startup's top 5 stories, tools, ideas, books delivered straight into your inbox, once a week. Take a look. Get this newsletter Your email By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy More from The Startup Follow Get smarter at building your thing. Follow to join The Startup's +8 million monthly readers & +760K followers. Martín Pettinati · Aug 10, 2020 🔶 Member-only Screwed Up A practical guide for deciding when to build, buy or adapt — The names Robertson, Allen, Parker, Philips and Goodwin may not mean anything to you, but they all played a part in building the world you live in. No,... Innovation 9 min read Share your ideas with millions of readers. Write on Medium Sergi Lehkyi · Aug 10, 2020 🔶 Member-only 5 Essential Data Vs You Have to Take Care Of Who starts good ends even better;) — I am pretty sure that on your data journey you came across some courses, videos, articles, maybe use cases where someone takes some data, builds a classification/regression... Data 5 min read The Single Most Important Self-Check in These Times of Upheaval Are you building a bridge to nowhere? — To say the world is changing would be an understatement. The raging-bull-loose-in-a-china-shop... Business 4 min read Emilee Lord · Aug 10, 2020 Contextualizing COVID-19 Data to Better Mitigate Reopening's Impact on Hispanic Communities There is no standardized method for reporting data on Hispanics in the United States, and many states did not even allow COVID-19 patients to... Hispanics 6 min read Moshe Kerbel · Aug 10, 2020 → Member-only Understanding the Weird Parts of Typescript Typescript, in spite of all its advantages, can be hard to swallow at first. Readability of Typescript code, is the main reason in my opinion, for developers being reluctant to give it a real shot. Let's take a look at this... Typescript 7 min read Read more from The Startup

**Recommended from Medium** 

me a bit

Ly Channa

John Rudd

not be the right fit

**WUNIFICA** 

INJECT

CodeRevGuru in Dev Genius

How to Ace Your First Code

Change at Your New Job

Install App As Admin Mac

DevNews: AWS 101—Basic

Mohamed Ayman

Techniques

Codegno... in Unification Foundat...

Configuration for Validator Nodes

SQL Injection (SQLI) - Manual

kindclever

Josh Bruce in 8fold 8fold UI Kit is starting to annoy Indifferent Access with Elixir/Phoenix App CQRS and why Repositories may Tagline Infotech LLP How can I choose a mobile application development company In the US?

Get started Sign In Q Search Leonardo Lima 123 Followers Computer Engineering student, Golang and Python developer, robotics and astronautics enthusiast, and open source defender Follow More from Medium 🚱 Clement Brian in Level Up Coding 6 Programming Languages With the Highest Jobs (and Those With the Lowest) Nigel in The Dev Project C++ Series: Constructors, and what happens when you don't implement a copy constructor The Educative T... in Dev Learning D... 5 reasons you should use C++ for digital signal processing M Manabendra Saha Coding is a Trap. Get Out. Help Status Writers Blog Careers Privacy Terms About Knowable