Como Compilar seus programas com CMake

O CMake tem sido bastante adotado pelo fato de ser multiplataforma e fácil de usar.

Dec 22, 2019

Nós já mostramos aqui um Tutorial Definitivo do GNU Autotools para Iniciantes bem como, um outro tutorial também de Como Criar um Makefile.

Mas o CMake tem sido bastante adotado pela comunidade/empresas pelo fato de ser multiplataforma e pelo fato de ser muito fácil de usar.

Introdução

CMake é um sistema multiplataforma para realizar geração automatizada. É comparável com o programa Unix Make no qual o processo de geração é, ao final, controlado pelos arquivos de configuração, no caso do CMake chamados de arquivos CMakeLists.txt.

Diferente de Make, ele não gera diretamente o software final, mas em vez disso gera arquivos de geração padrões (por exemplo, makefiles em Unix e projetos/espaços de trabalho no Visual C++ no Windows) os quais são usados de modo comum.

Isto permite que desenvolvedores familiarizados com um ambiente de desenvolvimento particular (tal como as várias IDEs) utilizem-o no modo padrão. É esta utilização do ambiente de geração nativo que distingue CMake dos outros sistemas mais conhecidos como o SCons.

O nome "CMake" é uma abreviação de "cross plataform make", ou em português make multiplataforma.

Instalação

Para instalar o CMake use o gerenciador de pacotes da sua distribuição, exemplos:

> Lembrando que você precisa também possuir um compilador instalado, exemplo:gcc/g++


```
sudo apt install cmake # Debian, Ubuntu, Linux Mint,...
sudo dnf install cmake # Fedora
sudo pacman -S cmake # Arch Linux, Manjaro,...
```

No Gentoo, Funtoo e similares, não precisa instalar, pois o CMake já é nativo desses sistemas.

Exemplo básico

Supondo que você possui o seguinte arquivo:

> Exemplo de construção de um código em C++

```
// main.cpp
#include <iostream>
int main(int argc, char** argv){
    std::cout << "Exemplo para teste com CMake" << std::endl;
    return 0;
}
```

Logicamente você poderia compilar com o GCC, the GNU Compiler Collection rodando o comando: `g++ main.cpp -o myexample` .

No entanto, para usar o CMake você primeiramente precisa criar um arquivo de nome: **CMakeLists.txt**: `vim CMakeLists.txt`(necessário escrever dessa forma, respeitando as letras em minúsculas e maiúsculas).

Dentro desse arquivo(para esse exemplo básico, e que serve até para projetos maiores) você precisará incluir 3 linhas:

- A primeira linha você vai informar a versão do seu CMake: `cmake_minimum_required(VERSION 3.10)` , consulte a versão que você possui para informá-la ou mesmo uma inferior a ela: `cmake --version`.
- Na segunda linha você deve informar o nome do seu projeto, exemplo: `project(MeuExemplo)`
- E na terceira linha informe o nome do binário final e do arquivo:

`add_executable(myexample main.cpp)`

> (se houver mais de um arquivo, exemplo se fosse diretamente pelo g++: `g++ main.cpp outro.cpp -o binario`) , no CMakeLists seria: `add_executable(binario main.cpp outro.cpp)` , assim como diretamente pelo compilador não precisa informar as bibliotecas(.h. .hh) , pois já fazem parte do includes, a não ser que não esteja.

```
// CMakeLists.txt
cmake_minimum_required(VERSION 3.10)
project(MeuExemplo)
add_executable(myexample main.cpp)
```

Após isso, basta criar um diretório os ficarão os arquivos do CMake e executar o comando `cmake`:

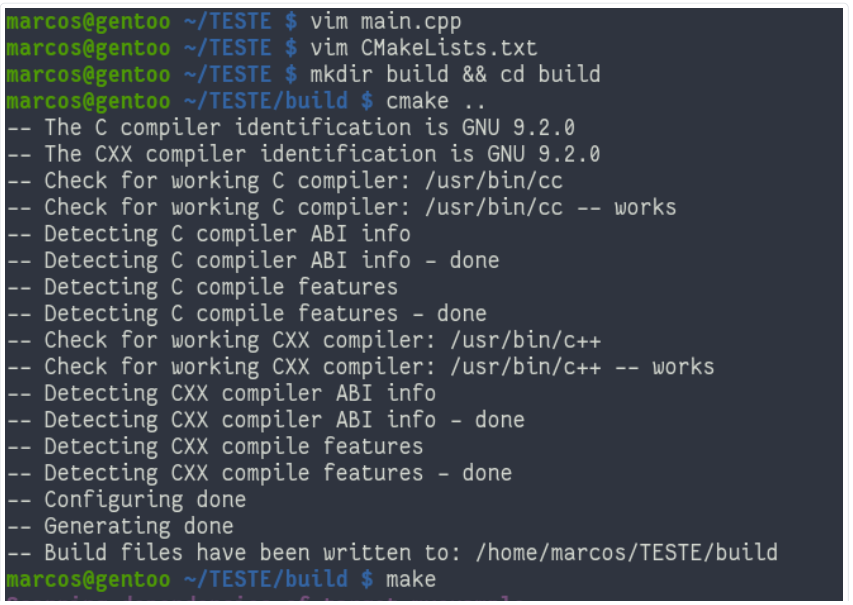
> Crie um diretório separado para ficar tudo organizado e em seguida execute o cmake apontando para o diretório imediatamente anterior "**..**"

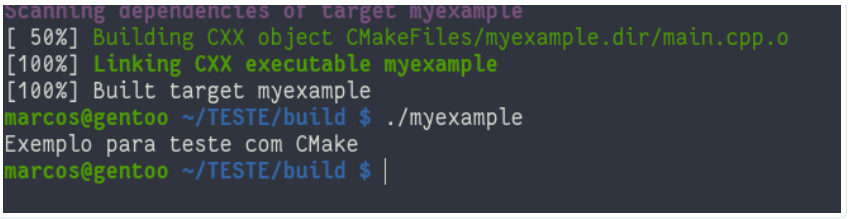

```
mkdir build && cd build
cmake ..
```

Isso gerou diversos arquivos de verificação do CMake e também o Makefile , agora basta compilar e testar o binário final:

```
make
./myexample
```

A saída será similar à imagem abaixo:





Mais informações

Existem mais coisas que você pode fazer dentro do CMakeLists.txt , como: controle de versão de projeto, binários, condições e entre outros, mas em resumo ele se resume à isso demonstrado aqui.

Caso você tenha interesse em saber mais, recomendo você ler o tutorial do próprio CMake clicando aqui

Links úteis:

- https://cmake.org/
- https://pt.wikipedia.org/wiki/CMake
- https://terminalroot.com.br/2019/12/como-criar-um-makefile.html

---

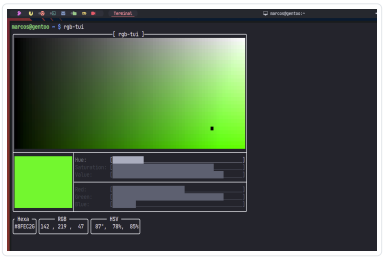
cmake cpp linguagemc compiladores gcc llvm clang make

Marcos Oliveira

Desenvolvedor de software

https://github.com/terroo

Artigos Relacionados

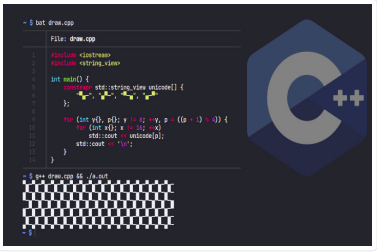


22 Dec 2021

Selecione Cores no Terminal com RGB-TUI C++

20 Dec 2021

Top 6 Linguagens de Programação para Aprender em 2022



16 Dec 2021

10 Exemplos de uso de string_view em C++

Site sobre C++, Sistemas Operacionais, Programação e Desenvolvimento Web. . Compra segura e conteúdo confiável com PayPal.

Conheça também

Blog do Edivaldo Linux no Café NewsTR Contribua English version