CMake – CTest

Kutatómunka Információs Eszközei

CTest

- A CMake képes arra, hogy unit test-eket buildeljen a fordítás mellett.

- Egy unit test egy olyan C++ projekt, amely által létrehozott futtatható állomány 0-t ad vissza, ha az adott teszt sikeres, és mást (pl.: -1) ha nem sikeres.

CTest – példa C++ unit test kód:

```
int main(){
  const double Iref = 0.65617436273150682981668121630094195846148449707466;
  const double I =
  integrate([](double x){ return std::exp(-x*x)*std::cos(x); }, 100, 0.0, 1.0);


  if(std::abs(I-Iref) < 1e-7){ return 0; }
  else{ return -1; }
}
```

CTest – példa CMake kód:

```
cmake_minimum_required(VERSION 3.0.0)
project(CTest_Test)
include(CTest)
enable_testing()
add_executable(CTest_Test main.cpp)

if(BUILD_TESTING)
  add_executable(test1 test1.cpp)
  add_test(NAME test1 COMMAND test1)
  add_executable(test2 test2.cpp)
  add_test(NAME test2 COMMAND test2)
endif(BUILD_TESTING)
```

Be kell include-olni a CTest funkcionalitást

És engedélyezni kell a tesztelést

CTest – példa CMake kód:

```
cmake_minimum_required(VERSION 3.0.0)
project(CTest_Test)
include(CTest)
enable_testing()
add_executable(CTest_Test main.cpp)

if(BUILD_TESTING)
  add_executable(test1 test1.cpp)
  add_test(NAME test1 COMMAND test1)
  add_executable(test2 test2.cpp)
  add_test(NAME test2 COMMAND test2)
endif(BUILD_TESTING)
```

Ez a normál futtatható alkalmazás

Ezek pedig a teszt futtathatók.

Minden teszthez tartozik egy végrehajtható (add_executable) és egy névvel ellátott teszt (add_test)

CTest használat:

- Használat parancssorból:
- Hozzunk létre egy build mappát a forrás fájlok mellett, lépjünk bele
- Hívjuk meg a cmake-t:
  `cmake ..`
- Build-eljük az alapértelmezett target-et:
  `cmake --build .`
- Hajtsuk végre az összes teszt-et (az adott konfigurációt meg kell adni a C kapcsolóval):
  `ctest -C Debug`
- Ha csak egy tesztet szeretnénk futtatni, akkor adjuk meg a nevét az R kapcsolóval:
  `ctest -R test2 -C Debug`

Automatizált műveletsorok CMake-el

Automatizált műveletsorok CMake-el

- A CMake segítségével automatizálhatjuk más programok egymásutáni végrehajtását is.

- Ez azért hasznos, mert a CMake követi, hogy melyik fájlok módosultak, és csak azokat a parancsokat futtatja újra, amelyek előfeltételéül megadott fájlok módosultak az utolsó futtatás óta.

Automatizált műveletsorok CMake-el

CMake pipeline:

- Itt lefordítunk egy C++ fájlt, amely egy adatfájlt fog előállítani (sin.dat)
- Futtatjuk az lefordított programot, abból előáll az adatfájl
- Meghívjuk a gnuplot-ot, hogy ábrázolja az adatfájlt,
  és csináljon belőle egy képet (sin.png)

CMake pipeline

```
cmake_minimum_required (VERSION 2.8.12)
project (CMakePipeline LANGUAGES CXX)
add_executable (CMakePipeline main.cpp)
add_custom_command(   COMMAND CMakePipeline
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.dat
                      DEPENDS CMakePipeline
                      COMMENT "Generating data set")
add_custom_target(data ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.dat)
find_package (Gnuplot REQUIRED)
add_custom_command(   COMMAND ${GNUPLOT_EXECUTABLE} ${PROJECT_SOURCE_DIR}/sin.plt
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.png
                      DEPENDS ${PROJECT_SOURCE_DIR}/sin.plt data
                      COMMENT "Generating plot")
add_custom_target(plot ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.png)
```

CMake pipeline

```
cmake_minimum_required (VERSION 2.8.12)
project (CMakePipeline LANGUAGES CXX)
add_executable (CMakePipeline main.cpp)
add_custom_command(   COMMAND CMakePipeline
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.dat
                      DEPENDS CMakePipeline
                      COMMENT "Generating data set")
add_custom_target(data ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.dat)
find_package (Gnuplot REQUIRED)
add_custom_command(   COMMAND ${GNUPLOT_EXECUTABLE} ${PROJECT_SOURCE_DIR}/sin.plt
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.png
                      DEPENDS ${PROJECT_SOURCE_DIR}/sin.plt data
                      COMMENT "Generating plot")
add_custom_target(plot ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.png)
```

Ez eddig a szokásos, lefordítunk egy C++ fájlt

CMake pipeline

```
cmake_minimum_required (VERSION 2.8.12)
project (CMakePipeline LANGUAGES CXX)
add_executable (CMakePipeline main.cpp)
add_custom_command(   COMMAND CMakePipeline
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.dat
                      DEPENDS CMakePipeline
                      COMMENT "Generating data set")
add_custom_target(data ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.dat)
find_package (Gnuplot REQUIRED)
add_custom_command(   COMMAND ${GNUPLOT_EXECUTABLE} ${PROJECT_SOURCE_DIR}/sin.plt
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.png
                      DEPENDS ${PROJECT_SOURCE_DIR}/sin.plt data
                      COMMENT "Generating plot")
add_custom_target(plot ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.png)
```

Itt megadhatunk egy tetszőleges parancsot, ami előállít valamilyen kimenetet.

Most az előző C++ programot hívjuk meg és a sin.dat fájlt állítjuk elő.

Ez a parancs függ (DEPENDS) a C++ program lefordításától, amit a project neve alapján kötünk össze

CMake pipeline

```
cmake_minimum_required (VERSION 2.8.12)
project (CMakePipeline LANGUAGES CXX)
add_executable (CMakePipeline main.cpp)
add_custom_command(   COMMAND CMakePipeline
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.dat
                      DEPENDS CMakePipeline
                      COMMENT "Generating data set")
add_custom_target(data ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.dat)
find_package (Gnuplot REQUIRED)
add_custom_command(   COMMAND ${GNUPLOT_EXECUTABLE} ${PROJECT_SOURCE_DIR}/sin.plt
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.png
                      DEPENDS ${PROJECT_SOURCE_DIR}/sin.plt data
                      COMMENT "Generating plot")
add_custom_target(plot ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.png)
```

A CMake belépési pontjai a targetek. Az itt megadott névre lehet hivatkozni, vagy hozzáadni az alapértelemzett targethez (ALL)

Itt is a DEPENDS-el adjuk meg, hogy mitől függ ez a target.

A CMake ki fogja keresni, hogy melyik parancsok kimenetében szerepel ez a fájl, és azokat meghívja.

CMake pipeline

```
cmake_minimum_required (VERSION 2.8.12)
project (CMakePipeline LANGUAGES CXX)
add_executable (CMakePipeline main.cpp)
add_custom_command(   COMMAND CMakePipeline
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.dat
                      DEPENDS CMakePipeline
                      COMMENT "Generating data set")
add_custom_target(data ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.dat)
find_package (Gnuplot REQUIRED)
add_custom_command(   COMMAND ${GNUPLOT_EXECUTABLE} ${PROJECT_SOURCE_DIR}/sin.plt
                      WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
                      OUTPUT ${CMAKE_BINARY_DIR}/sin.png
                      DEPENDS ${PROJECT_SOURCE_DIR}/sin.plt data
                      COMMENT "Generating plot")
add_custom_target(plot ALL DEPENDS ${CMAKE_BINARY_DIR}/sin.png)
```

Végül az előállított adatfájlra meghívjuk a gnuplot-ot hasonlóan az előzőekhez.

A find package gondoskodik arról, hogy megkeresse a gnuplot elérési útvonalát.

CMake pipeline

- Használat parancssorból:
- Hozzunk létre egy build mappát a forrás fájlok mellett, lépjünk bele
- Adjuk hívjuk meg a cmake-t:
  `cmake ..`

- Build-eljük az alapértelmezett target-et:
  `cmake --build .`

CMake pipeline

- Ha a gnuplot-ot nem találja a cmake, akkor adjuk át az elérési útját, pl.:

```
cmake -DGNUPLOT_EXECUTABLE="C:\Program Files\gnuplot\bin\gnuplot.exe" ..
```

- Ha csak egy adott targetet akarunk buildelni:

```
cmake --build . --target data
```