RTミドルウェアによるロボットプログラミング技術

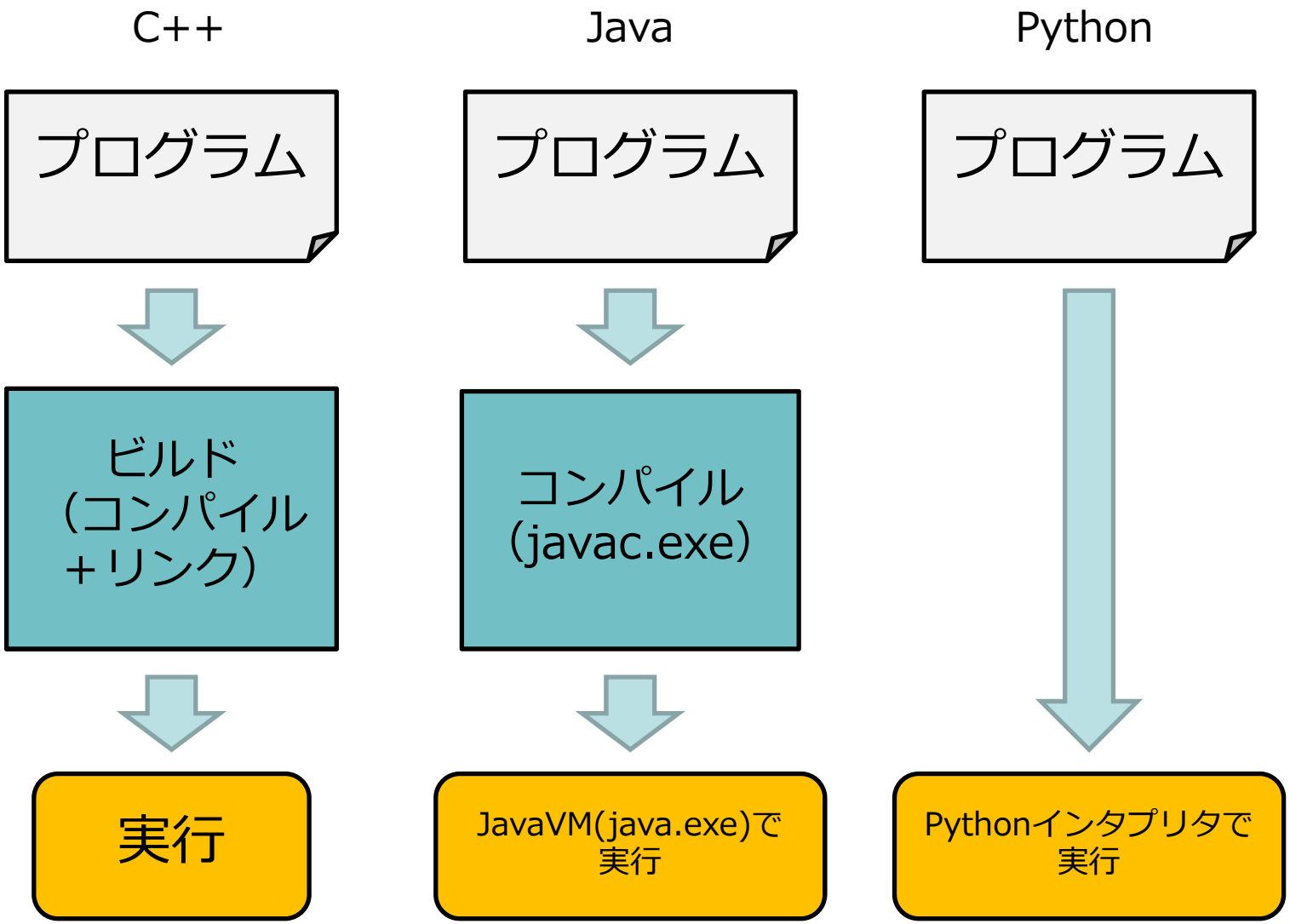
2．プログラミングの基礎

概要

1. プログラミングの基礎
2. Linuxでのプログラミング
3. Windowsでのプログラミング

目標:
1. LinuxおよびWindowsでの開発手法(主にC++)を学ぶ
2. CMakeを利用して、同じプログラムをLinuxとWindowsとでコンパイルして動作させる。

プログラミングの流れ

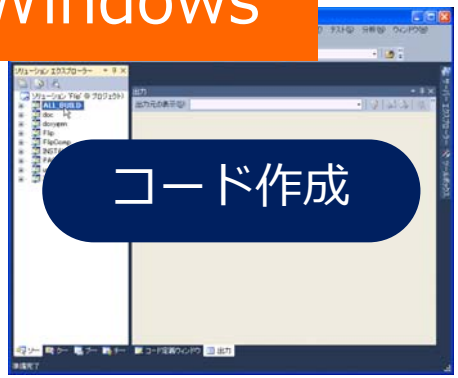

C++
Java
Python
プログラム
プログラム
プログラム
ビルド
(コンパイル
+リンク)
コンパイル
(javac.exe)
実行
JavaVM(java.exe)で
実行
Pythonインタプリタで
実行

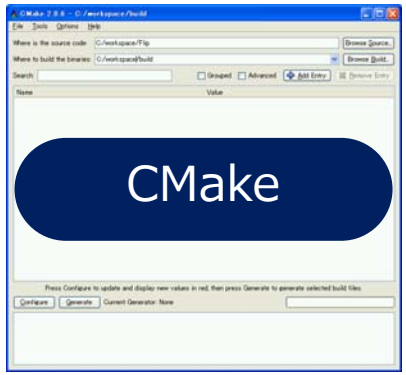
CMake

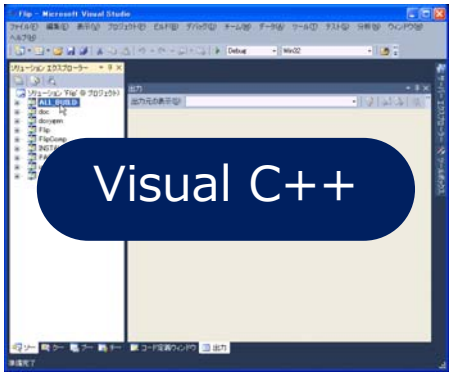
- コンパイラに依存しないビルド自動化のためのフリーソフトウェア
- 様々なOS上の様々な開発環境用ビルドファイルを生成することができる
  - Linux では Makefileを生成
  - Windows ではVCのプロジェクトファイルを生成
- 最近のオープンソースソフトウェアではCMakeでビルドするようになっているものが多数。

プログラム作成の流れ

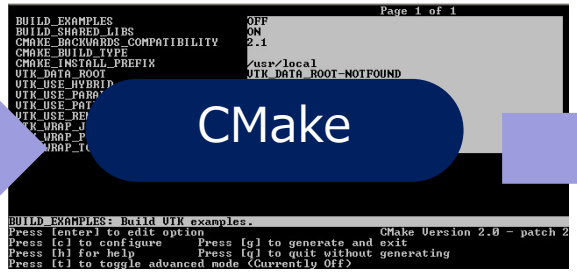
Windows

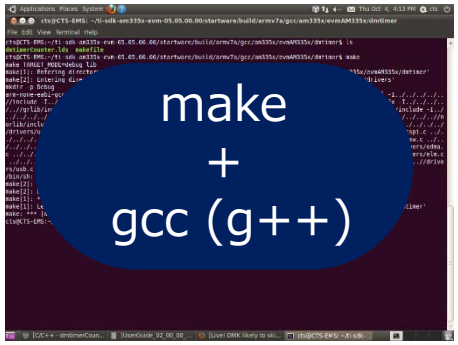






Linux





コンポーネントの
仕様の入力

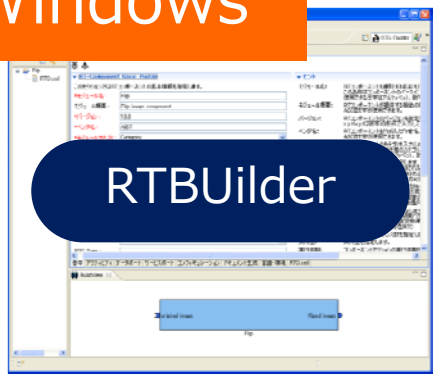
VCプロジェクトファイル
またはMakefileの生成

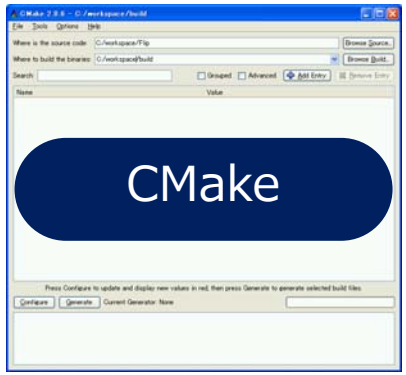
実装およびコンパイル
実行ファイルの生成

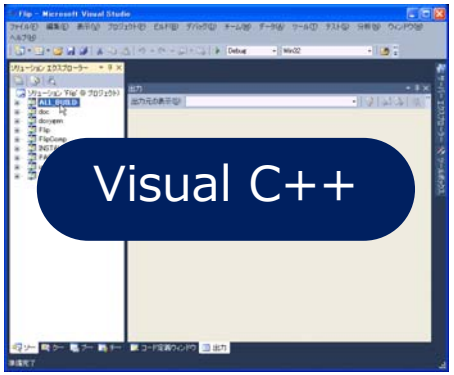
途中まで流れは同じ、コンパイラが異なる

コンポーネント作成の流れ

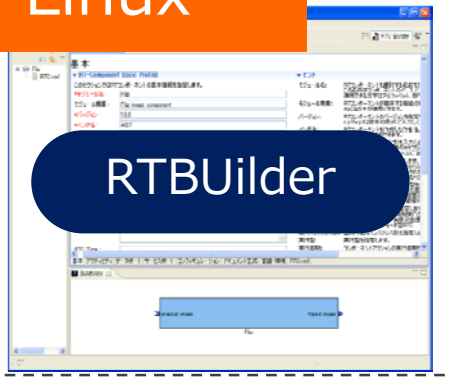
Windows

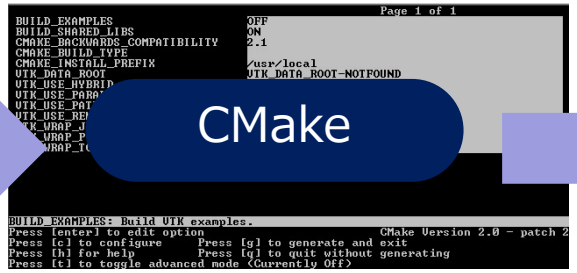


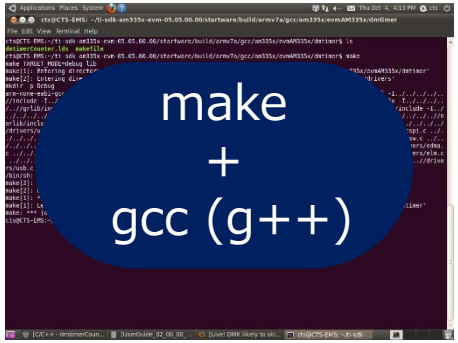




Linux







コンポーネントの
仕様の入力

VCプロジェクトファイル
またはMakefileの生成

実装およびコンパイル
実行ファイルの生成

途中まで流れは同じ、コンパイラが異なる

演習：CMakeを使ってみよう

- CMakeを使って同じプログラムをLinuxとWindowsの両方でビルドしてみる。

- 手順：
  - サンプルプログラムをダウンロード
  - CMakeLists.txtを編集
  - cmake (cmake-gui)
  - make or VC++ でコンパイル
  - 実行

ソフトウェアのインストール (Linux)

- openrtm.orgの
  - 「ダウンロード」→「C++」→「1.1.2」
- pkg_install_ubuntu.sh をダウンロード
- sudo sh pkg_install_ubuntu.sh を実行
- cmakeもインストール

Linuxパッケージ

現在のところ以下のディストリビューション・バージョンでパッケージを提供しています。
以下で配布しているインストールスクリプトを利用すれば、必要なパッケージを一括でインストールすることができます。

ディストリビューション・バージョン	一括インストールスクリプト
Ubuntu 12.04 (precise) i386/amd64 Ubuntu 14.04 (trusty) i386/amd64 Ubuntu 15.10 (wily) i386/amd64 Ubuntu 16.04 (xenial) i386/amd64	pkg_install_ubuntu.sh
Debian 7.0 (wheezy) i386/amd64 Debian 8.0 (jessie) i386/amd64	pkg_install_debian.sh

【使用コマンド一覧】
wget: ファイルなどをダウンロード
sudo: 管理者権限で実行
sh: シェルコマンド
apt-get: パッケージインストール

```
$ wget http://svn.openrtm.org/OpenRTM-aist/tags/RELEASE_1_1_2/OpenRTM-aist/build/pkg_install_ubuntu.sh
$ sudo sh pkg_install_ubuntu.sh –c
$ sudo apt-get install cmake
```

ダウンロード

Linux

- ブラウザからダウンロード または端末 “terminal” を開いて wgetで取得＆展開
- 以降の操作は terminal で行うので開いたままに

【使用コマンド一覧】
mkdir: フォルダ(ディレクトリ)を作成
wget: ファイルなどをダウンロード
unzip: ZIPファイルを展開
cd: ディレクトリに移動
ls: ファイル一覧を表示

```
$ mkdir work
$ wget http://openrtm.org/openrtm/sites/default/files/6135/arm2dof_ver001.zip
--2016-11-05 22:57:38--
        :
$ unzip arm2dof_ver001.zip
Archive:  arm2dof_ver001.zip
   creating: arm2dof/
  inflating: arm2dof/arm2dof.cpp
  inflating: arm2dof/CMakeLists.txt
$ cd arm2dof/
$ ls
CMakeLists.txt  arm2dof.cpp
$
```

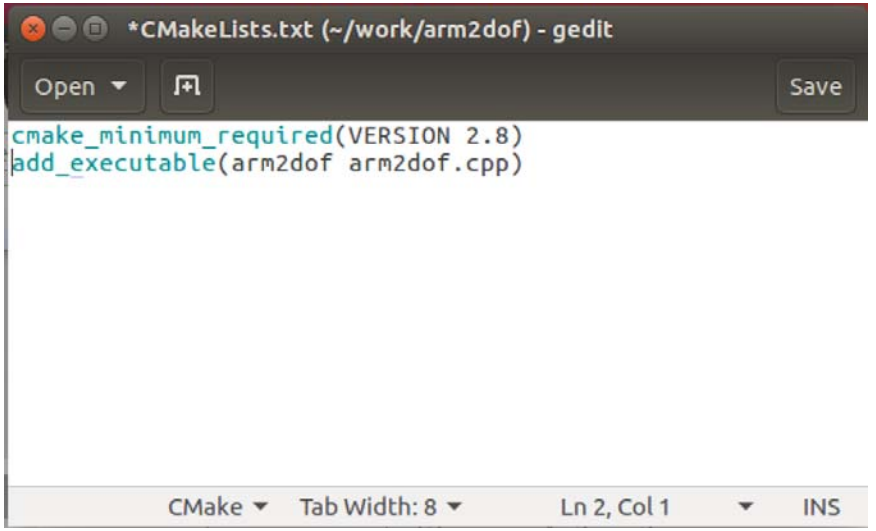
CMakeLists.txtの編集

Linux

- CMakeLists.txt をgedit(エディタ)で開く
- 2行コメントイン
  - 行頭の '#' を削除
- 保存・終了

```
$ gedit CMakeLists.txt

または
$ vi CMakeLists.txt
または
$ emacs CMakeLists.txt
```



geditの編集画面

add_executable(arm arm2dof.cpp)
の行は、

- 実行ファイル(executable) arm2dof を作成せよ
- そのためのソースコードは arm2dof.cpp である

ということを意味している。

cmake & make & 実行

Linux

- build ディレクトリを作成
- cmake .. を実行
- make を実行
- arm2dofが生成される
- arm2dofを実行

```
n-ando@Ubuntu1604-64:~/work/arm2dof/build$ mkdir build

n-ando@Ubuntu1604-64:~/work/arm2dof/build$ cmake ..
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0

    : 中略

-- Configuring done
-- Generating done
-- Build files have been written to: /home/n-ando/work/arm2dof/build

n-ando@Ubuntu1604-64:~/work/arm2dof/build$ make
Scanning dependencies of target arm2dof
[ 50%] Building CXX object CMakeFiles/arm2dof.dir/arm2dof.cpp.o
[100%] Linking CXX executable arm2dof
[100%] Built target arm2dof

n-ando@Ubuntu1604-64:~/work/arm2dof/build$ ls
arm2dof  CMakeCache.txt  CMakeFiles  cmake_install.cmake
Makefile

n-ando@Ubuntu1604-64:~/work/arm2dof/build$ ./arm2dof
pos (x, y): -1, 1                ==> angle (th0, th1): 0, 0
pos (x, y): -0.5, 1              ==> angle (th0, th1): 0, 0
pos (x, y): 0, 1  ==> angle (th0, th1): 0, 0
pos (x, y): 0.5, 1               ==> angle (th0, th1): 0, 0
pos (x, y): 1, 1  ==> angle (th0, th1): 0, 0
n-ando@Ubuntu1604-64:~/work/arm2dof/build$
```

【使用コマンド一覧】

cmake: コマンド

make: Makefileに基づいてコンパイル・リンクするためのコマンド

arm2dof: 今回作成する実行ファイル

【ディレクトリ指定】

. or ./ : 現在のディレクトリ（カレントディレクトリ）

.. or ../ : 現在のディレクトリの一つ上のディレクトリ（親ディレクトリ）

ソフトウェアのインストール（Windows）

- http://bit.ly/2fNLE6a にアクセス
- 以下をインストール
  - Python
  - OpenRTM-aist
  - PyYAML
  - Cmake
  - Doxygen
  - TeraTerm

ダウンロード

Windows

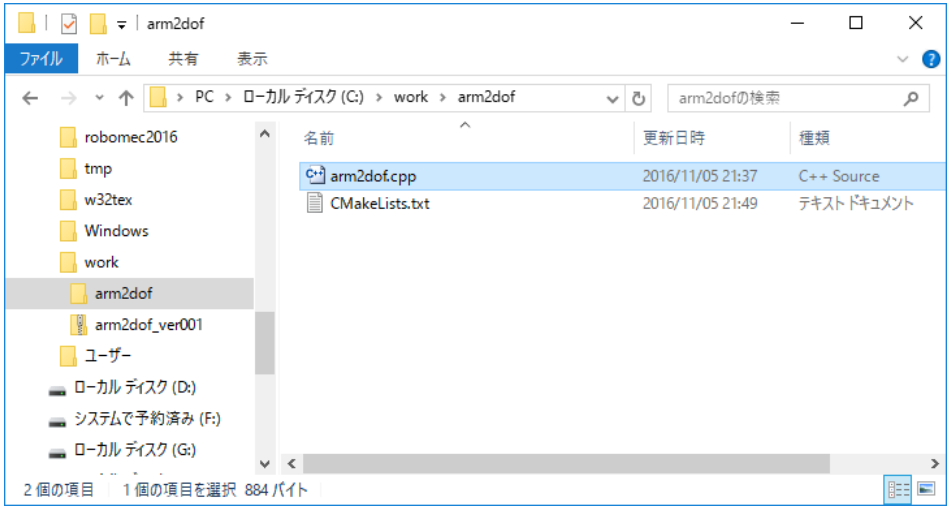
http://bit.ly/2fpwDon

Windows

- ブラウザからダウンロード
- ZIPを展開
- arm2dofフォルダの下に
  - arm2dof.cpp
  - CMakeLists.txt

  の2つのファイルが展開される

11月10日(木)	
10:00 -11:00	1. コース概要 (1)ロボットシステムプログラミングの現状 (2)ロボットOS・ミドルウェア (3)RTミドルウェア(RTM)を用いたロボット開発 資料: 161110-01.pdf
11:00 -12:00 13:00-14:00	2. プログラミングの基礎 (1)プログラミングの基礎 (2)Linuxでのプログラミング (3)Windowsでのプログラミング サンプルコード: arm2dof_ver001.zip



CMakeLists.txtの編集

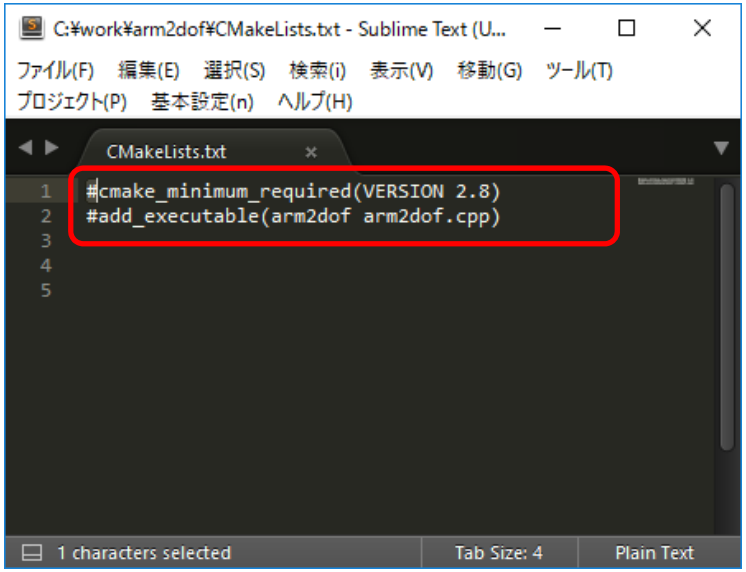
Windows

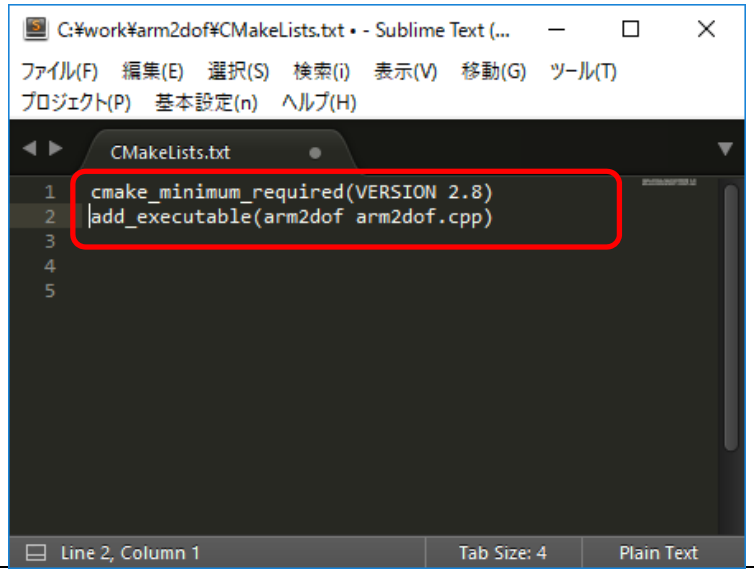
- CMakeLists.txt をエディタで開く
- 2行コメントイン
  - 行頭の ‘#’ を削除

add_executable(arm arm2dof.cpp)
の行は、

- 実行ファイル(executable) arm2dof を作成せよ
- そのためのソースコードは arm2dof.cpp である

ということを意味している。



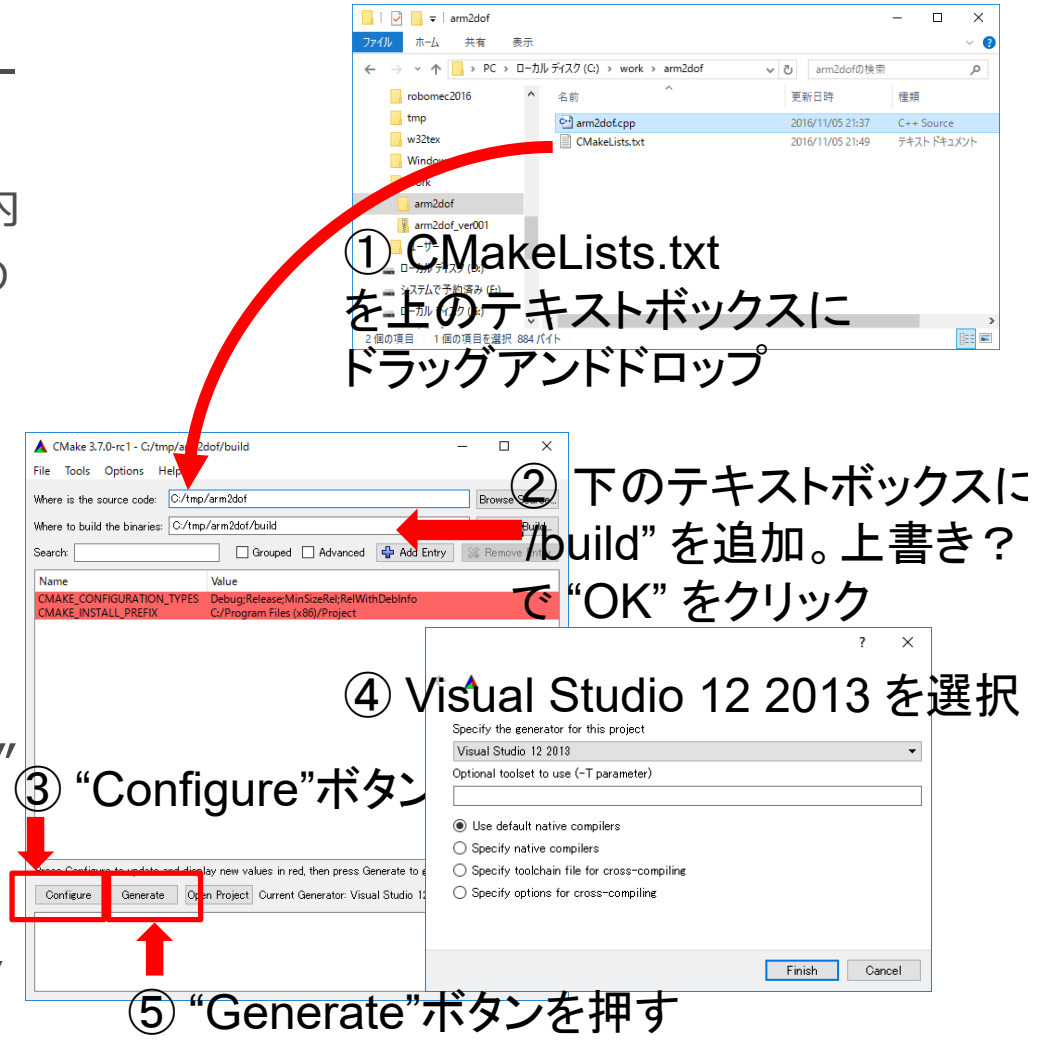


cmake-gui

Windows

- スタートメニューからcmake-gui を起動 

  - スタートメニュー “cmake”内
  - 検索窓でcmakeと入力するのが早い
- CMakeLists.txtを受けのテキストBOXにDnD
- “Configure”ボタンを押下
- ダイアログでVisual Studio 12 2013を選択
- 下の窓に “Configuring done” ならOK
- “Generate”ボタンを押下
- 下の窓に “Generating done” と出ていれば完了



コンパイル（ビルド）

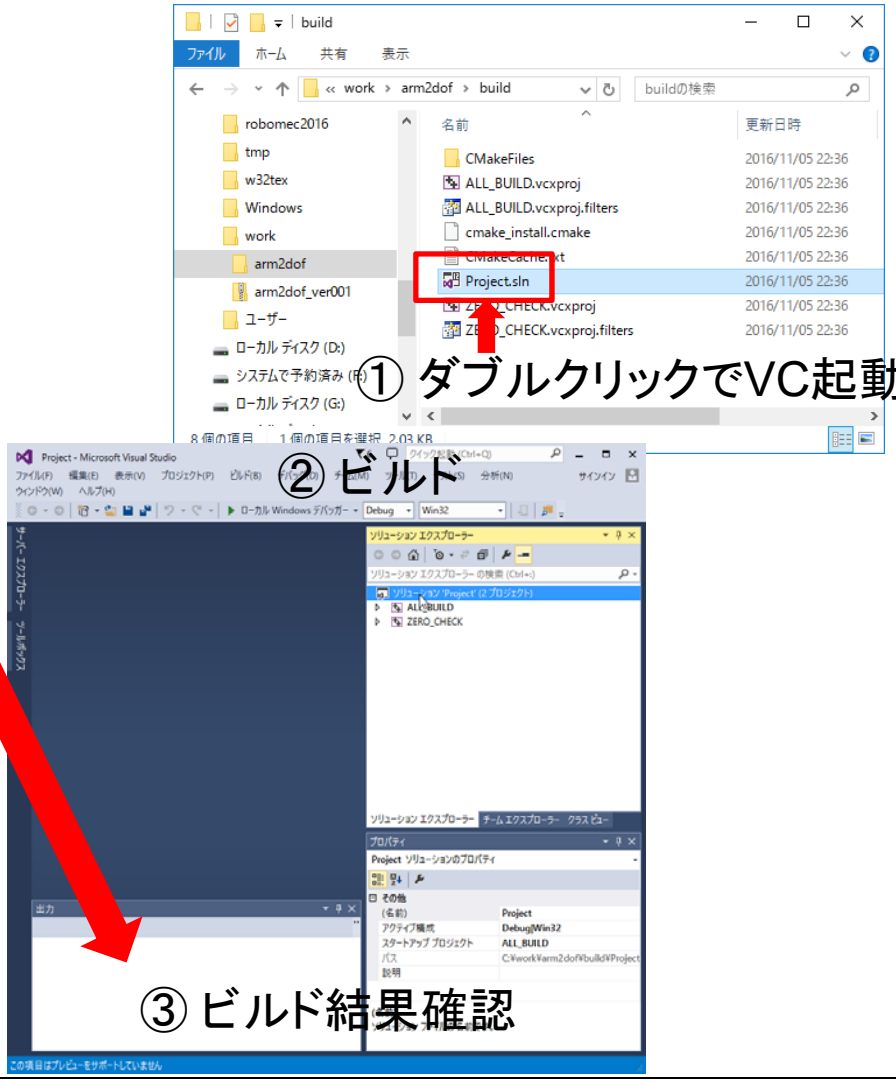
Windows

- Build フォルダ内の“Project.sln”をダブルクリック
- Visual C++ 2013が起動
- “ビルド”→”ソリューションのビルド”でコンパイル

ビルド: 2 正常終了、0 失敗、0 更新不要、1 スキップ

と出ていれば正常終了

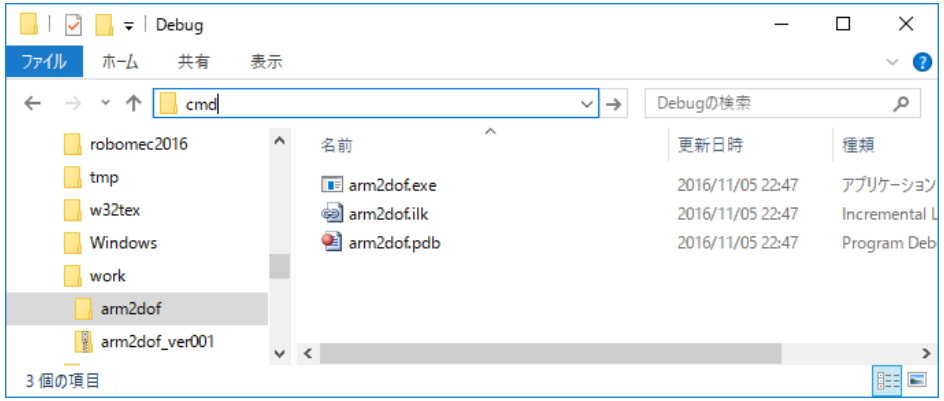
- arm2dof¥build¥Debug の下に arm2dof.exe ができている

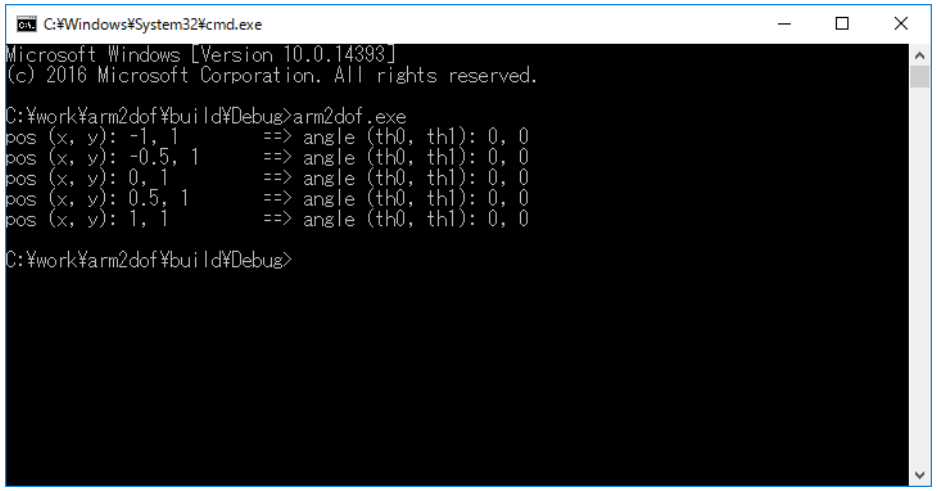


実行

Windows

- コマンドプロンプトを起動
  - Explorerのアドレスバーに“cmd”と入力しEnter
- プロンプトで“arm2dof.exe”と入力
- 結果が表示される。
  - ダミーコードなので、angle = (0, 0) でしか表示されない
  - 2日目に練習問題として実装してもらいます。





まとめ

- CMakeを利用すると、同じソースコードをWindowsでもLinuxでコンパイルできる
  - CMakeLists.txt をcmake (または cmake-gui)で処理
  - Windowsでは プロジェクトファイルとソリューションファイル
  - Linuxでは Makefile を作成
  - それぞれの方法でビルド