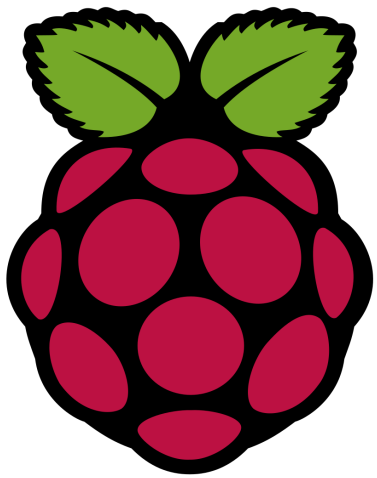
Introduction au kit de développement logiciel Pico SDK



December 5, 2022

Un système similaire à celui du langage Bash

- Aucun typage (ce qui est différent du typage statique en C et du typage dynamique en Python)
- Variables définies par défaut
- Lecture des variables par expansion, par exemple en Bash

```
MY_VAR=Hello
echo MY_VAR  # Affiche 'MY_VAR'
echo $MY_VAR # Afficher 'Hello'
```

Un système similaire à celui du langage Bash

```
set(EXECUTABLE_NAME MyExecutable)
set(EXECUTABLE_SOURCES "")
list(APPEND EXECUTABLE_SOURCES main.c)
list(APPEND EXECUTABLE_SOURCES aux.c)
add_executable(${EXECUTABLE_NAME} ${EXECUTABLE_SOURCES})
```

La commande set pour définir ou modifier une variable

```
set(<variable> <value>... [PARENT_SCOPE])
set(<variable> <value>... CACHE <type> <docstring> [FORCE])
set(ENV{<variable>} [<value>])
```

La visibilité des variables

```
project/
├── CMakeLists.txt
└── src/
    └── CMakeLists.txt
```

CMakeLists.txt

```
add_subdirectory(src)
message(WARNING ${MY_VAR})
```

CMakeLists.txt

```
set(MY_VAR "Hello")
```

La visibilité des variables

```
$ cmake -B build
CMake Warning at CMakeLists.txt:5 (message):


-- Configuring done
-- Generating done
-- Build files have been written to: ...
```

La visibilité des variables

```
project/
├── CMakeLists.txt
└── src/
    └── CMakeLists.txt
```

CMakeLists.txt

```
add_subdirectory(src)
message(WARNING ${MY_VAR})
```

CMakeLists.txt

```
set(MY_VAR "Hello"
PARENT_SCOPE)
```

La visibilité des variables

```
$ cmake -B build
CMake Warning at CMakeLists.txt:5 (message):
Hello

-- Configuring done
-- Generating done
-- Build files have been written to: ...
```

La commande `list` pour traiter une variable comme une liste

```
Reading
  list(LENGTH <list> <out-var>)
  list(GET <list> <element index> [<index> ...] <out-var>)
  list(JOIN <list> <glue> <out-var>)
  list(SUBLIST <list> <begin> <length> <out-var>)

Search
  list(FIND <list> <value> <out-var>)

Modification
  list(APPEND <list> [<element>...])
  list(FILTER <list> {INCLUDE | EXCLUDE} REGEX <regex>)
  list(INSERT <list> <index> [<element>...])
  list(POP_BACK <list> [<out-var>...])
  list(POP_FRONT <list> [<out-var>...])
  list(PREPEND <list> [<element>...])
  list(REMOVE_ITEM <list> <value>...)
  list(REMOVE_AT <list> <index>...)
  list(REMOVE_DUPLICATES <list>)
  list(TRANSFORM <list> <ACTION> [...])

Ordering
  list(REVERSE <list>)
  list(SORT <list> [...])
```

Déclarer une liste

```
set(MY_LIST "Hello" "There !" "Hello There !")
message(${MY_LIST})
message("${MY_LIST}")

set(MY_LIST Hello There ! Hello There !)
message(${MY_LIST})
message("${MY_LIST}")
```

Déclarer une liste

```
$ cmake -B build
HelloThere !Hello There !
Hello;There !;Hello There !
HelloThere!HelloThere!
Hello;There;!;Hello;There;!
-- Configuring done
-- Generating done
-- Build files have been written to: ...
```

La sous-commande APPEND de list

```
set(MY_LIST "Hello" "There !" "Hello There !")
list(APPEND MY_LIST "Hi !")
message(${MY_LIST})
message("${MY_LIST}")
```

La sous-commande APPEND de list

```
$ cmake -B build
HelloThere !Hello There !Hi !
Hello;There !;Hello There !;Hi !
-- Configuring done
-- Generating done
-- Build files have been written to: ...
```

La variable spéciale CMAKE_MODULE_PATH

```
project/
├── CMakeLists.txt
└── tool/
    └── MyModule.cmake
```

CMakeLists.txt

```
include(MyModule)
```

La variable spéciale CMAKE_MODULE_PATH

```
$ cmake -B build
CMake Error at CMakeLists.txt:5 (include):
  include could not find requested file:

    MyModule


-- Configuring incomplete, errors occurred!
```

La variable spéciale CMAKE_MODULE_PATH

```
project/
├── CMakeLists.txt
└── tool/
    └── MyModule.cmake
```

CMakeLists.txt

```
list(APPEND
CMAKE_MODULE_PATH
${PROJECT_SOURCE_DIR}/tool)
include(MyModule)
```

Définir une fonction

```
function(my_function A B C)
    message("${A} ${B} ${C}")
endfunction()

my_function("Hello" "there" "!")
```

Définir une fonction

```
$ cmake -B build
Hello there !
-- Configuring done
-- Generating done
-- Build files have been written to: ...
```

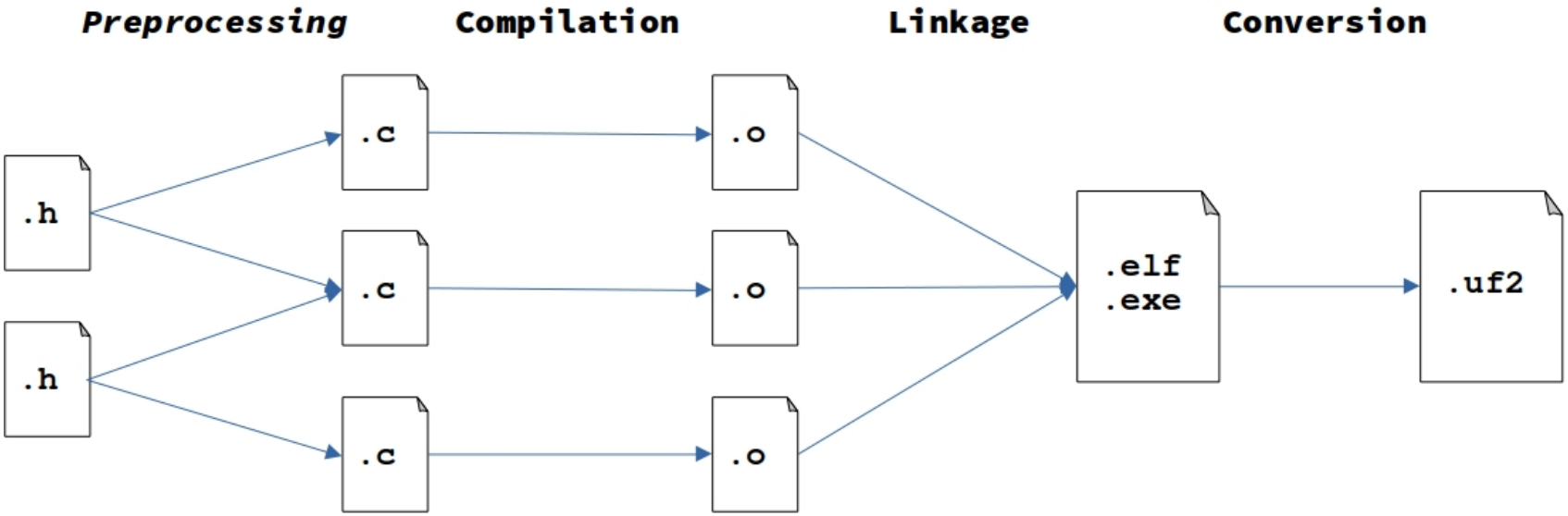
TP : Créer un module

Écrivez un module qui défini une fonction de votre choix.
Pour tester, le module, écrivez un `CMakeLists.txt` qui inclus le module et appelle la fonction.

Définir une fonction

- Le SDK se présente comme **un projet CMake**
- Le dépôt SDK se trouve sur GitHub (possibilité d'utiliser FetchContent)
- Le SDK fourni **des modules**, **des fonctions CMake** (qui se trouvent dans le dossier `external/` du dépôt) et **des bibliothèques** (à linker à vos exécutables)
- La difficulté est **d'inclure le SDK dans votre projet CMake**

Une étape supplémentaire après la production de l'exécutable




```
pico_add_extra_outputs(MyExecutable)
```

Initialisation du SDK

```
...

FetchContent_Declare(pico-sdk
    GIT_REPOSITORY https://github.com/raspberrypi/pico-sdk
    GIT_SUBMODULES_RECURSE OFF)

...

include(pico_sdk_import)

...

project(TechTheTaste_LowLevel LANGUAGES C CXX ASM)
```

Activer et désactiver des fonctionnalités du SDK

```
# `printf()` écrit sur la liaison USB
pico_enable_stdio_usb(MyExecutable ON)

# `printf()` écrit sur la liaison UART
pico_enable_stdio_uart(MyExecutable ON)
```

TP: Projet CMake + Raspeberry Pico

Ce TP est à réaliser en groupe de 2 ou 3 personnes.

- Créez un dépôt GitHub pour le TP
- Initialiser le dépôt avec l'arborescence suivante

```
├── CMakeLists.txt
├── log/
│   └── CMakeLists.txt
└── init/
    └── CMakeLists.txt
```

Les fichiers doivent être **vides**

- Télécharger le dépôt sur l'ordinateur de chaque membre puis créer une branche par membre

TP: Projet CMake + Raspeberry Pico

Répartissez les tâches suivantes entre les membres du groupes :

- Dans le dossier `init/`, écrivez une fonction qui initialise la liaison USB
- Dans le dossier `log/`, écrivez une fonction qui permet de d'écrire sur la liaison USB
- A la racine du projet, écrivez une fonction `main` qui utilise les deux fonctions précédentes

Une contrainte supplémentaire est ajoutée : **vous ne devez pas écrire en dehors du répertoire qui vous est réservé**. Vous ne devez pas non plus faire de *commits* en dehors de votre branche. En revanche, vous avez le droit de fusionner les branches avec la branche `main`