Exercise 3 : Cross compilation & Makefiles

Exercise 3.1 : First try - KISS

Cross compiling is simple in the sense that g++, in our case, is exchanged with another g++ that supports the desired target, e.g. arm-rpizw-g++. That being said lets revisit exercise 2.2 in which a simple makefile was made.

Start by making a copy of this makefile called makefile.target and change it such that it uses the target compiler. Having done that invoke it using make.

Consider:

- **Do you have to do something special to invoke this particular makefile?**
- **At this point we have two makefiles in the same dir. How does this present a problem in the current setup and how are you forced to handle it?**
  - In this current setup, it's not a problem, as Make will, by default, go for files named Makefile or makefile. We can specify what file to Make, by using -f or --file

We changed the compiler to target our Raspberry pi

```
CXX=arm-rpizw-g++
```

Because we are using a nonstandard name for our makefile, we specify the makefile name with the '-f' or '--file' option:

```
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
3.1$ ls
 hello.cpp  '#makefile.target#'   Makefile.target   Makefile.target~
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
3.1$ make -f Makefile.target
arm-rpizw-g++ -o RUN hello.cpp
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
3.1$ ls
 hello.cpp  '#makefile.target#'   Makefile.target   Makefile.target~   RUN
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
```

Moving executable to Raspberry pi:

```
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
3.1$ scp RUN root@10.9.8.2:/ISU/
RUN                                           100%   17KB   1.4MB/s   00:00
stud@stud-virtual-machine:~/ISU/i3isu_e2018_hal9000/Lecture1_exercises/Exercise
3.1$
```

Execute on Raspberry pi:

```
root@raspberrypi0-wifi:/ISU# ls
Exercise3.2  RUN
root@raspberrypi0-wifi:/ISU# ./RUN
Hello world
FFS
```

Exercise 3.2 : The full Monty - bye bye KISS

At this point we want to develop the final makefile that handles all the issues encountered in one go.

Take your starting point in the listing 3.1 and finalize the missing parts such that we get a

makefile that attains the desired functionality:

```
 7  # Making for host
 8  # > make ARCH=host
 9  ifeq (${ARCH},host)
10  CXX=g++
11  BUILD_DIR=build/host
12  endif
13
14  # Making for target
15  # > make ARCH=target
16  ifeq (${ARCH},target)
17  CXX=arm-rpizw-g++
18  BUILD_DIR=build/target
19  endif
20
21
22  $(EXE): $(DEPS) $(OBJECTS)   # << Check the $(DEPS) new dependency
23    $(CXX) $(CXXFLAGS) -o $@ $(OBJECTS)
24
25  # Rule that describes how a .d (dependency) file is created from a .cpp
      file
26  # Similar to the assigment that you just completed %.cpp -> %.o
27  ${BUILD_DIR}/%.d: %.cpp
28    $(CXX) -MT$(@:.d=.o) -MM $(CXXFLAGS) $^ > $@
29
30  ifneq ($(MAKECMDGOALS),clean)
31  -include $(DEPS)
32  endif
```

The following Makefile is created using the given listing 3.1, previous makefiles and with the help of fellow students:

```
#Name the executeable file the same as the folder:
BIN_NAME := $(notdir $(shell pwd))
#Name of output:
OUTPUT   := build
#Source files created:
SOURCES  := $(subst ./,,$(shell find -name '*.c' -o -name '*.cpp'))
#Object files created:
OBJECTS  := $(patsubst %.cpp, ${OUTPUT}/%.o, ${SOURCES})
#Dependencies created:
DEP      := $(OBJECTS:%.o=%.d)
#Include dependencies
-include $(DEP)

#Define what compiler to use:
ifeq (${ARCH},host)
CXX=g++
BUILD_DIR=build/host
endif

ifeq (${ARCH},target)
CXX=arm-rpizw-g++
BUILD_DIR=build/target
endif

#Common CFLAGS:
CFLAGS      := -c
CXXFLAGS=-I
#Common LFLAGS:
LDFLAGS  := -lm
#Create a folder for the build:
$(shell mkdir -p -m a=rwx ${OUTPUT})

all: $(BIN_NAME)
```

```
${BIN_NAME}:${OBJECTS}
    @echo "[Linking] "$@
    @${CXX} ${LDFLAGS} ${OBJECTS} -o ${OUTPUT}/$@
    @echo "done, executeable is located @ "${OUTPUT}/$@

.PHONY : clean
clean:
    @rm -rf $(OUTPUT)
    @echo "Done."

${OUTPUT}/%.o : %.cpp
    @echo "[Compiling] "$@
    @mkdir -p -m a=rwx ${dir $@}
    @${CXX} ${CFLAGS} $^ -o $@

help:
    @echo Just write make or clean

ifneq (${MAKECMDGOALS},clean)
-include ${DEPS}
endif
```

The makefile is now designed to either build to host or to target. The choice is made by invoking:

```
make ARCH=host // will build for host system
make ARCH=target // will build for Raspberry pi
```

The makefile is build and moved to the Raspberry pi:

Execution on the Raspberry pi:

Filer

Listing3.1.png	60,6 KB	2018-09-05	Brian Nymann
auto_make_test.png	87,2 KB	2018-09-10	Brian Nymann
3.1_2.png	36,8 KB	2018-09-10	Brian Nymann
3.1_2.png	36,8 KB	2018-09-10	Brian Nymann
file_transfered_and_run.png	11,7 KB	2018-09-10	Brian Nymann
make_the_file.png	69,3 KB	2018-09-10	Brian Nymann
move_to_rpi.png	19,7 KB	2018-09-10	Brian Nymann
run_on_rpi.png	8,19 KB	2018-09-10	Brian Nymann