mauriciopoppe / _README.md

Last active 5 months ago

Generic Makefile example for a C++ project

_README.md

Generic makefile example for a C++ project

If you need a small makefile introduction/reference you can take a look at my notes
https://www.mauriciopoppe.com/notes/os/bin/make/

Project structure

```
. project
├── Makefile
├── build
├── include
│   └── project
│       └── Vector.hpp
└── src
    ├── Vector.cpp
    └── main.cpp
```

```
$ make
Creating directories
Compiling: src/Vector.cpp -> build/Vector.o
c++  -std=c++11 -Wall -Wextra -g -I include/ -I /usr/local/include -MP -MMD -c src/Vector.cpp -o build/Vector
Compiling: src/main.cpp -> build/main.o
c++  -std=c++11 -Wall -Wextra -g -I include/ -I /usr/local/include -MP -MMD -c src/main.cpp -o build/main.o
Linking: build/bin/runner
c++ build/Vector.o build/main.o -o build/bin/runner
Making symlink: runner -> build/bin/runner

$ ./runner
(3,3)

$ make clean
Deleting runner symlink
Deleting directories
```

Generic makefile stolen from https://github.com/mbcrawfo/GenericMakefile

include_project_Vector.hpp

```
1  #ifndef PROJECT_VECTOR_
2    #define PROJECT_VECTOR_
3
4  class Vector {
5    double x,y;
6  public:
7    Vector();
8    Vector(double,double);
```

```
  Vector operator+(const Vector &other) const;
  void print();
};

#endif
```

src_Main.cpp

```
#include "project/Vector.hpp"

int main () {
  Vector a(1,2), b(2, 1);
  a = a + b;
  a.print();
  return 0;
}
```

src_Vector.cpp

```
#include <iostream>

#include "project/Vector.hpp"

Vector::Vector() {
  x = y = 0.0;
}

Vector::Vector(double a, double b) {
  x = a;
  y = b;
}

Vector Vector::operator+(const Vector &other) const {
  return Vector(x + other.x, y + other.y);
}

void Vector::print() {
  std::cout << "(" << x << "," << y << ")" << std::endl;
}
```

~Makefile

```
CXX ?= g++

# path #
SRC_PATH = src
BUILD_PATH = build
BIN_PATH = $(BUILD_PATH)/bin

# executable #
BIN_NAME = runner

# extensions #
SRC_EXT = cpp

# code lists #
# Find all source files in the source directory, sorted by
# most recently modified
SOURCES = $(shell find $(SRC_PATH) -name '*.$(SRC_EXT)' | sort -k 1nr | cut -f2-)
# Set the object file names, with the source directory stripped
# from the path, and the build path prepended in its place
OBJECTS = $(SOURCES:$(SRC_PATH)/%.$(SRC_EXT)=$(BUILD_PATH)/%.o)
# Set the dependency files that will be used to add header dependencies
DEPS = $(OBJECTS:.o=.d)

# flags #
COMPILE_FLAGS = -std=c++11 -Wall -Wextra -g
```

```
INCLUDES = -I include/ -I /usr/local/include
# Space-separated pkg-config libraries used by this project
LIBS =

.PHONY: default_target
default_target: release

.PHONY: release
release: export CXXFLAGS := $(CXXFLAGS) $(COMPILE_FLAGS)
release: dirs
	@$(MAKE) all

.PHONY: dirs
dirs:
	@echo "Creating directories"
	@mkdir -p $(dir $(OBJECTS))
	@mkdir -p $(BIN_PATH)

.PHONY: clean
clean:
	@echo "Deleting $(BIN_NAME) symlink"
	@$(RM) $(BIN_NAME)
	@echo "Deleting directories"
	@$(RM) -r $(BUILD_PATH)
	@$(RM) -r $(BIN_PATH)

# checks the executable and symlinks to the output
.PHONY: all
all: $(BIN_PATH)/$(BIN_NAME)
	@echo "Making symlink: $(BIN_NAME) -> $<"
	@$(RM) $(BIN_NAME)
	@ln -s $(BIN_PATH)/$(BIN_NAME) $(BIN_NAME)

# Creation of the executable
$(BIN_PATH)/$(BIN_NAME): $(OBJECTS)
	@echo "Linking: $@"
	$(CXX) $(OBJECTS) -o $@

# Add dependency files, if they exist
-include $(DEPS)

# Source file rules
# After the first compilation they will be joined with the rules from the
# dependency files to provide header dependencies
$(BUILD_PATH)/%.o: $(SRC_PATH)/%.$(SRC_EXT)
	@echo "Compiling: $< -> $@"
	$(CXX) $(CXXFLAGS) $(INCLUDES) -MP -MMD -c $< -o $@
```