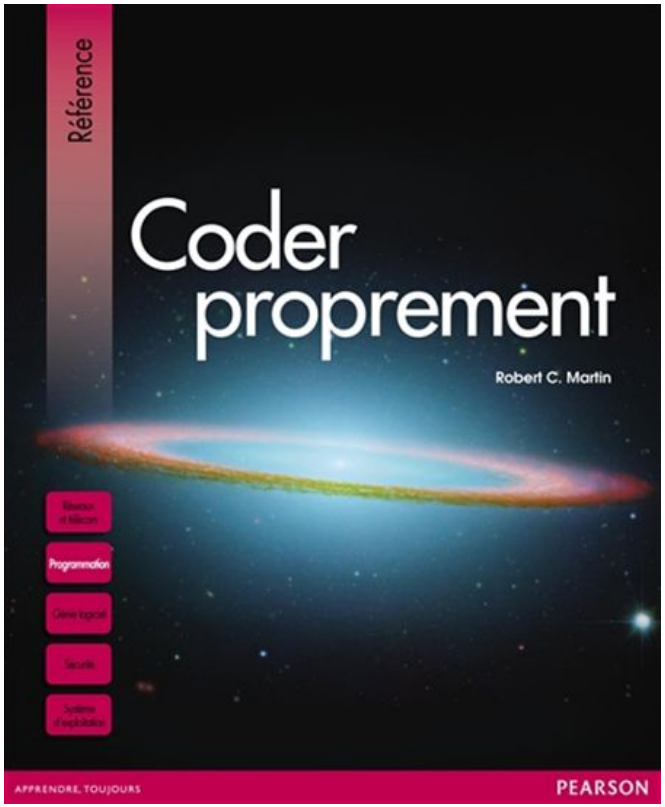
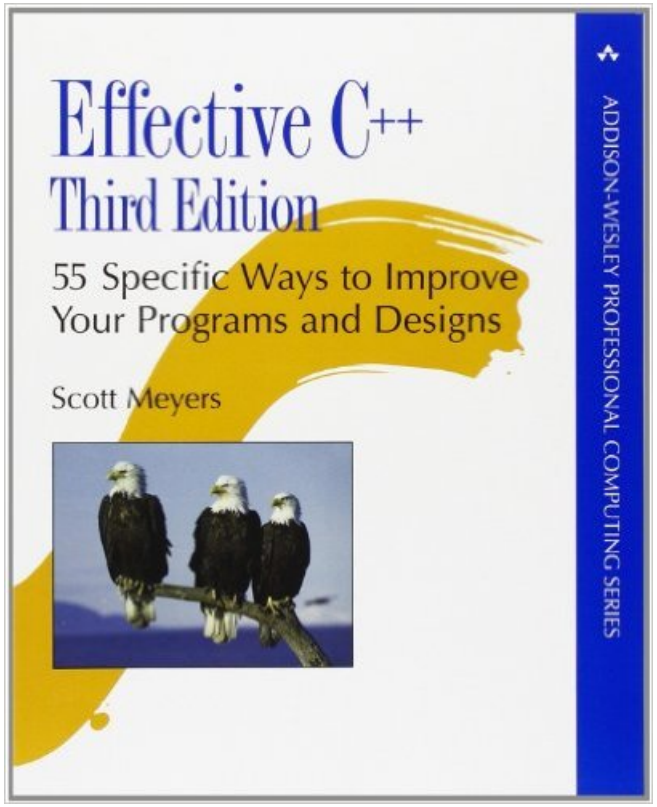
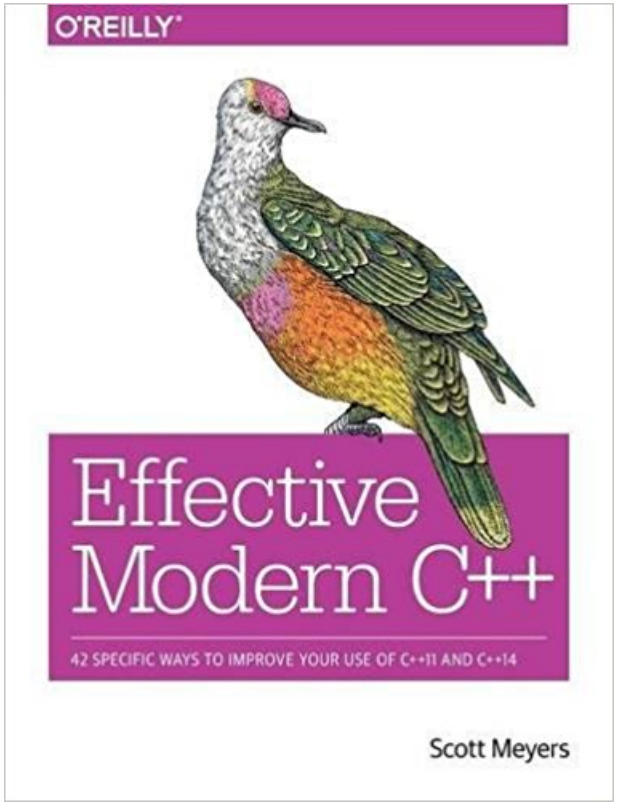
François Corneau-Tremblay


Référence
Coder proprement
Robert C. Martin
Programmation
APPRENDRE, TOUJOURS
PEARSON


Effective C++
Third Edition
55 Specific Ways to Improve Your Programs and Designs
Scott Meyers
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


O'REILLY
Effective Modern C++
42 SPECIFIC WAYS TO IMPROVE YOUR USE OF C++11 AND C++14
Scott Meyers

Makefile

Makefiles

- Qu’est-ce que c’est?
  - Ensemble de règles
  - Construction d’objet et de librairie
  - Usage généralisé
- Exemple d’utilisation :
  - https://github.com/torvalds/linux/blob/master/Makefile

L’intérêt des Makefile

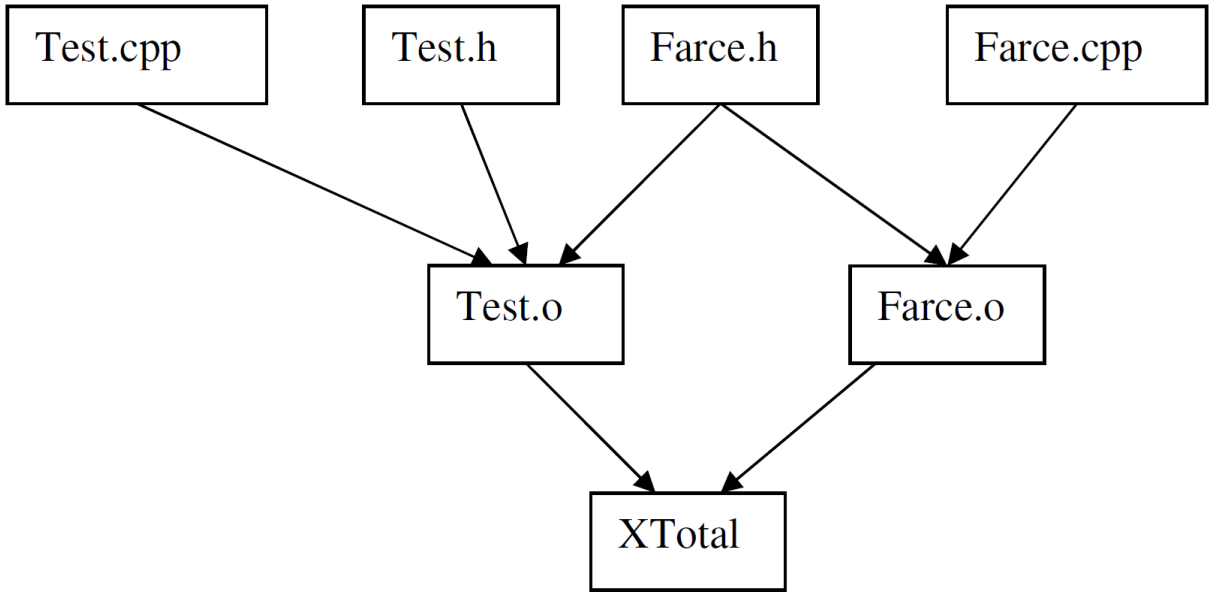
- Automatiser l’exécution de tâches
- Optimiser le temps d’exécution
- Ex :
  - g++ -o FichExe Fic1.cpp Fic2.cpp Fic3.cpp ……. Fic99.cpp

Contenu d’un Makefile

- Cibles : Label donné à une règle d’exécution. Représente un objectif ou un fichier
- Dépendances : Règle(s) dont la cible courant dépend.
- Commandes : Ce qui doit être exécuté par la cible
- Variables

Exemple 1:

- Demo1\Demo_Makefile\makefile_base



Variables

- Syntaxe :
  - Déclaration -> NOM_VARIABLE = VALEUR
  - Appel -> $NOM_VARIABLE
- Exemples :
  - CXX = g++
  - CXXFLAGS = -Os –Wall
  - OPT = -pedantic
- Variables implicites :
  - C++ -> CXX et CXXFLAGS
  - C -> CC et CFLAGS

Variables automatiques

- $@ : nom de la cible associée à la règle
- $< : nom de la première dépendance
- $? : liste des dépendances (séparées par un espace blanc) qui sont plus récentes que la cible
- $^ : liste de toutes les dépendances (séparées par un espace blanc)
- $* : nom du fichier sans suffixe

Opérateurs

- Syntaxe générale
  - OPERATEUR (arg1, arg2)
  - OPERATEUR ‘arg1’, ‘arg2’
  - OPERATEUR ‘’arg1’’, ‘’arg2’’
  - OPERATEUR ‘arg1’, ‘’arg2’’
  - OPERATEUR ‘’arg1’’, ‘arg2’
- Ifeq / else / endif
- Ifneq, ifdef, ifndef

Exemple 2:

- Demo_Makefile\makefile
- Farce.o: Farce.cpp Farce.h
- 	$(CXX) $(CXXFLAGS) $(OPT) -c $< -o $@
- $< : Farce.cpp
- $@ : Farce.o
- $? : Farce.cpp Farce.h
- $^ : Farce.cpp Farce.h
- $* : Farce

Exemple 3:

- Demo_Makefile\makefile2
- Utilisation des variables pour nommées :
  - Cibles
  - Dépendances
  - Commandes

Règles génériques

- Cible agissant en fonction de l’extension des fichiers
- Syntaxe : %.o : %.cpp
- Crée une règle qui exécutera la même commande pour tous les fichiers .o
- ‘%’ remplace le nom du fichier

Exemple 4:

- Demo_Makefile\makefile3
- $(objets): Farce.h
  - Chaque fichier .o dépend du fichier .cpp portant le même nom. Cependant le fichier Test.cpp dépend aussi de Farce.h alors il faut l'ajouter manuellement à la liste des dépendances.

Documentation complète

- http://www.gnu.org/software/make/

MSYS2

Procédure d’installation

- http://www.msys2.org/
- Télécharger la version x86_64
- Pour les plus suspicieux, vérifier le SHA256 de l’installeur
- Lancer MSYS2
- Exécuter : pacman –Syu
- Relancer MSYS2
- Exécuter : pacman –Su
- Mingw 32 et 64 bit (GCC et G++)
  - pacman -S mingw-w64-x86_64-gcc
- Make
  - pacman -S make

Outils

- Mingw 32 et 64 bit (GCC et G++)
  - pacman -S mingw-w64-x86_64-gcc
- Make
  - pacman -S make