

# Notes diverses

Michel Billaud ([michel.billaud@u-bordeaux.fr](mailto:michel.billaud@u-bordeaux.fr), [michel.billaud@laposte.net](mailto:michel.billaud@laposte.net))

9 janvier 2022

## Table des matières

1	Licence	1
2	Configurer CodeBlocks	2
3	Une vieille conjecture	2
4	Makefiles pour un projet en C avec un sous-projet	2
5	Sockets IPv4 et IPv6 en C	2
6	“Lambdas” en C	3
7	“Factory” en C	3
8	À propos des Makefiles	3
9	Polymorphisme en C	3
10	Génération des permutations	4
11	Vi, comment s’en sortir ? (et l’utiliser un peu)	4
12	Les tris	4
13	Réseau	4
14	Développer à partir des tests	5
15	Exercices “somme et moyenne d’un tableau”	5
16	Bases markdown	5

## 1 Licence



Cette collection de notes est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions 2.0 France.

- Les notes sont publiées dans <https://www.mbillaud.fr/notes/>
- Sources dans <https://github.com/MichelBillaud/notes-diverses>

## 2 Configurer CodeBlocks

Sur le Web, certains cours recommandent aux débutants en C/C++ d'utiliser l'IDE CodeBlocks pour leurs premiers pas en C ou C++

Cette note n'est pas là pour critiquer ce choix, ni les cours en question - malgré le mal que j'en pense - mais pour aider à le configurer correctement pour débiter. Ici on montre comment faire pour que le compilateur

- affiche les avertissements,
- vérifie que le code se conforme à une version précise du langage (de préférence, le standard le plus récent).

## 3 Une vieille conjecture

Janvier 2022. Il y a très longtemps (vers 88 je crois) je m'étais posé une question que je n'avais pas résolue. Ce n'était pas du tout mon domaine, et ça me paraissait plus dur que prévu. Apparemment ça a intéressé des gens (combinatoire des mots) et c'est toujours une question ouverte.

Si vous voulez tenter votre chance...

Document : [conjecture.html](#)

Todo : insérer la biblio des articles qui en causent.

## 4 Makefiles pour un projet en C avec un sous-projet

Quand un projet atteint une certaine taille, il arrive qu'on puisse en isoler une "bibliothèque" (sous-projet) qu'on espère réutiliser dans d'autres projets. Ici on montre comment organiser et compiler un projet (en C) qui contient un sous-projet. Et surtout, on explique les **Makefiles** qui vont avec.

Document : [makefile-sous-projet.html](#)

## 5 Sockets IPv4 et IPv6 en C

Janvier 2022. Les cours de programmation réseau/C utilisent obstinément une vieille API, en ignorant complètement IPv6, et les améliorations apportées au langage C depuis sa première normalisation en 1989.

On essaie de faire un peu mieux.

Document : sockets-ipv6.html

## 6 “Lambdas” en C

Novembre 2021. Comment simuler des “lambdas” en C dans un programme d’énumération des mots de Dyck (systèmes de parenthèses bien formés), qui se base sur un algorithme récursif, fondé sur la grammaire  $S \rightarrow \epsilon \| aSbS$ .

Document : enumeration-mots-dyck-en-c.html

## 7 “Factory” en C

Novembre 2021. On montre comment mettre en oeuvre en C le patron de conception “factory”, dans lequel une fonction est utilisée pour fabriquer des objets de types distincts.

Document : factory-en-c.html

C’est une continuation de celui sur le polymorphisme en C

## 8 À propos des Makefiles

Novembre 2021. Récupération de quelques notes d’explication sur l’utilisation des Makefiles sur des projets de petite taille en C. Ces notes ont été écrites pour un cours C/Système en 2018.

### Documents

- makefile-gen-dependances.html Un Makefile doit indiquer les dépendances entre les sources et les fichiers produits. La note montre comment l’option `-MMD` de `gcc` évite d’avoir à le faire manuellement.
- makefile-options.html Comment indiquer des options spécifiques pour *certaines* cibles d’un `Makefile`.
- makefile-projet.html Quand un projet contient de nombreux fichiers sources, il est intéressant de produire les fichiers intermédiaires et les exécutables dans des répertoires séparés des sources.

## 9 Polymorphisme en C

Novembre 2021. On montre sur un exemple détaillé comment réaliser du *polymorphisme* en C

Le code suivant :

```
Animal * animals[] = {
    (Animal *) new_Dog("Medor"),
    (Animal *) new_Fish("Yellow"),
    (Animal *) new_Dog("Rex")
};

for (int i = 0; i < 3; i++) {
```

```

        Animal_Talk(animals[i]);
        Animal_Feed(animals[i], "the kibbles"); // croquettes
    }

```

exécute, lors des appels d'`Animal_Talk` et `Animal_Feed`, du code différent, qui dépend du type effectif (`Dog` ou `Fish`) des objets. C'est réalisé grâce à une “table de fonctions virtuelles” propres à chaque type.

Document : [polymorphisme-en-c.html](#)

## 10 Génération des permutations

Novembre 2021. Cette note explique en détail un algorithme classique pour passer d'une permutation, par exemple (3, 5, 2, 4, 1) à la suivante dans l'ordre lexicographique (3, 5, 4, 1, 2).

La technique présentée peut se généraliser à des énumérations d'autres types (note à venir).

Le code est fourni en Fortran 95 (ça m'amusait de l'apprendre, j'en étais resté à la version 77) et en C.

Document : [permutation-suivante.html](#)

## 11 Vi, comment s'en sortir ? (et l'utiliser un peu)

L'éditeur de textes vi est apparu en 1976, il est présent sur la majorité des systèmes Unix (vim, version améliorée). C'est parfois le seul éditeur installé au départ, et souvent l'éditeur de textes par défaut. On peut en préférer d'autres, mais il est important de savoir en sortir si on est tombé dedans, et de connaître quelques commandes de bases pour l'utiliser, même si on référerait autre chose.

Document : [vi-utilisation-basique.html](#)

## 12 Les tris

D'où viennent les idées sur les algorithmes de tri “naïfs” qu'on présente souvent aux débutants : tri par sélection, tri par insertion, tri à bulles ?

Document : [idees-tris-naifs.html](#)

## 13 Réseau

- Explication de la notion de trame, d'adresse (MAC), et de communication sur un réseau local.

Document : [reseau-local.html](#)

- Notions de routage dans un réseau (interconnexion de réseaux locaux). Adresses IP, préfixes et masques, configuration des interfaces, définition des routes.

Document : routage.html

## 14 Développer à partir des tests

Quand on programme, on se trompe souvent. Il est intéressant de prévoir des tests automatisés dès le début.

Illustration avec C.

Document : dev-tests-c.html

## 15 Exercices “somme et moyenne d’un tableau”

Exercices très simples qu’on donne souvent à écrire aux débutants. Le problème est qu’ils sont souvent mal posés, sur le plan pédagogique.

On présente une formulation de ces exercices (pour C), et on explique les problèmes courants avec les autres formulations.

Document : exo-somme-tableau.html

## 16 Bases markdown

Le strict minimum, pour les débutants.

Document : bases-markdown.html