Notes diverses

Michel Billaud (michel.billaud@laposte.net)

15 janvier 2023

Table des matières

1 Licence

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

2 Réalisation d’un pipeline en C

La réalisation d’un mini-*shell* (interprète de commandes) est un projet classique de programmation système en C.

Dans sa version la plus simpliste, un shell est une boucle qui

- affiche une chaîne d’invite (*prompt*),
- lit **une** commande,
- lance son exécution,
- et recommence.

La réalisation d’un tel programme n’est pas très compliquée.

Là où ça se gâte un peu, c'est si on veut exécuter des "pipelines" de commandes, c'est-à-dire plusieurs commandes dont la sortie standard est redirigée vers l'entrée de la suivante, comme dans

```
ls -l | grep -v ^d | more
```

La difficulté est essentiellement d'utiliser correctement les $n-1$ tuyaux qui interviennent dans un "pipeline" de n commandes, en les ouvrant ou fermant ni trop tôt, ni trop tard. Tous en évitant les fuites de descripteurs, etc.

Document : pipeline

3 Méthodologie de la programmation, tests, langage C

Le langage C n'est pas jeune, a plein de défauts, et est souvent très mal enseigné, surtout au regard des enjeux actuels : produire du code qui n'a pas trop d'erreurs.

Pour cela, il convient de sensibiliser les débutants qui découvrent la programmation en C. Un point qui est très souvent négligé, c'est l'idée de tester systématiquement le code que l'on écrit.

Mieux :

- d'**automatiser** les tests, pour qu'ils s'exécutent à chaque modification des sources, et pas seulement quand on n'aura que ça à faire d'y penser ;
- d'écrire les tests **avant** le code. Ca permet de réfléchir à ce que le code est censé faire, avant dêtre mentalement encombré par les détails de comment on envisage de le réaliser.

Pour enseigner ça, il n'est pas utile de montrer un "framework de test" à des débutants. Les bibliothèques industrielles c'est très utile pour les professionnels, mais là on s'adresse à des débutants qui sont déjà largement perdus dans les bases de C. Quand ils les maîtriseront ils apprendront à s'en servir si jamais ils en ont besoin. Et ils apprendront d'autant plus vite qu'ils auront maîtrisé les bases de la programmation, sans être ralentis par l'apprentissage d'une usine à gaz.

Bref, ce document montre

- ce que fait `assert` ;
- comment on l'utilise dans le cadre d'exercices de programmation ;

sur un exemple classique : quelques exercices sur les listes chaînées.

Document : methodo-c-tests-listes

4 Conjecture sur les mots

Il y a très longtemps (vers 1988), je m'étais posé un petit problème, que je n'ai pas réussi à résoudre. Rien d'étonnant, c'était pas dans mon domaine, et je n'y

ai travaillé que mollement. Donc j'ai laissé tomber rapidement, mais après avoir appelé à l'aide sur Usenet, qui ne manque pas de gens plus compétents.

> Given a word w, if for each letter x occurring in w, there exists non-trivial morphism f_x such that the word obtained by erasing all the occurrences of x in x is a fixed point of f_x, then there exists a non-trivial morphism f such that w is a fixed point of f.

Et bon, surprise, on est en 2022, mais apparemment c'est toujours pas résolu, et il y a des gens qui publient sur le sujet.

Document : conjecture

5 Arbres binaires quasi-complets

Suite à un exercice présenté sur Twitter, on veut connaitre la taille du sous-arbre gauche d'un arbre binaire de recherche quasi-complet à N sommets.

Quasi-complet : tous les niveaux de l'arbre sont remplis à part (éventuellement) le dernier, qui est rempli à partir de la gauche.

Note : je garde le terme "complet" pour ceux qui sont complètement complets.

Document : arbres-binaires

6 Génération en Verilog

Dans un épisode précédent on a vu comment obtenir un additionneur 2 x 4 bits + retenues à partir de 4 additionneurs 2 x 1 bit + retenues, en les listant explicitement.

Avec l'instruction `generate` de Verilog, nous allons éviter de construire ce type de code grâce à une boucle de génération, au lieu de faire du copier-coller-modifier.

Document : géneration-verilog

7 Description structurelle d'un additionneur en Verilog

On montre comment

- décrire un circuit en Verilog selon le style structurel, c'est-à-dire en représentant un module comme un assemblage de sous-modules et de portes logiques ;
- écrire des tests pour vérifier le bon fonctionnement, en simulant l'envoi de valeurs sur les entrées.

Exemples :

- un demi-additionneur formé d'une porte "et" et d'un "ou-exclusif",
- additionneur composé de deux demi-additionneurs et d'une porte "ou",
- additionneur 2 x 4 bits.

Document : additionneur-verilog

8 Vérifier le résultat des élections

Sur les réseaux sociaux, on trouve régulièrement des messages promouvant l'usage des machines à voter, ou du vote en ligne.

Pour faire bref, je suis contre les deux, mais ce n'est pas le point. L'objet de cette note c'est de réfuter l'affirmation à propos du vote "papier" :

> Ok, on peut surveiller le vote et le dépouillement (en étant présent dans les bureaux) , mais pour la totalisation nationale, on ne peut pas faire faire confiance.

en montrant

- où on peut obtenir les données de vote des bureaux
- comment vérifier la totalisation.

sur l'exemple du premier tour de la présidentielle 2022.

Pour ça, on verra comment

- analyser les données du fichier qu'on récupère au format CSV
- écrire un petit programme Python qui fait la totalisation.

Pourquoi Python ? Parce que c'est un langage de programmation

- à portée du non-professionnel (il y a une initiation à Python au Lycée, voire au collège)
- qui va assez bien pour faire ça. C'est un programme "one-shot" dont on ne se resservira pas tel quel, la vitesse de calcul n'a pas d'importance.

Document : calculs-elections.html

9 Permutation qui ordonne un tableau, en C

On veut trouver la permutation qui ordonne un tableau.

Exemple : pour le tableau

```
int array[9] = { 66, 11, 44, 22, 88, 55, 77, 99, 33};
```

la suite d'indices `{1, 3, 8, 2, 5, 0, 6, 4, 7}`

parce que `array[1]=11, array[3]=22, array[8]=33`, etc.

1. C'est facile à faire avec `qsort_r` qui est une extension GNU.
2. Un bricolage permet de le faire avec `qsort`, mais ça donne du code non réentrant.
3. On montre comment adapter un algo de tri en fonction de tri "réentrante" paramétrée par une fonction de comparaison.

Document : calculs-elections.html

10 Bibliothèque dynamique sous Linux, CodeBlocks

Dans de nombreux cours de programmation en C on demande aux élèves

- de travailler avec CodeBlocks (CB),
- d'utiliser une bibliothèque "maison" qu'on me fournit.

Je montre comment

- fabriquer une bibliothèque dynamique
- comment configurer CB pour faire fonctionner un projet qui utilise une telle bibliothèque.

Document : bib-dynamique-linux.html

11 Corrigé sujet 7 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33193/download :

- conversion d'un entier en liste de bits
- insertion dans une liste ordonnée

Document : corrige-22-NSI-07.html

12 Corrigé sujet 7 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33202/download :

- indice d'un élément dans une liste
-

Document : corrige-22-NSI-08.html

13 Corrigé sujet 6 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33193/download :

- recherche de la valeur et l'indice du maximum d'une liste
- recherche d'une sous-chaîne.

Document : corrige-22-NSI-06.html

14 Corrigé sujet 5 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33190/download :

- minimum et maximum d'une liste
- construction d'un paquet de cartes

Document : corrige-22-NSI-05.html

15 Corrigé sujet 4 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33187/download :

- encodage par différence d'une suite de nombres
- affichage d'un arbre d'expression arithmétique sous forme parenthésée.

Document : corrige-22-NSI-04.html

16 Corrigé sujet 3 épreuves pratiques NSI

Sujet dans https://eduscol.education.fr/document/33184/download :

- encodage par différence d'une suite de nombres
- affichage d'un arbre d'expression arithmétique sous forme parenthésée.

Document : corrige-22-NSI-03.html

17 Corrigé second sujet épreuves pratiques NSI

Janvier 2022.

Sujet dans https://eduscol.education.fr/document/33181/download :

- moyenne avec coefficients
- triangle de pascal (programme à trous)

Document : corrige-22-NSI-02.html

18 Corrigé du premier sujet d'épreuves pratiques NSI

Le sujet se trouve sur la page https://eduscol.education.fr/2661/banque-des-epreuves-pratiques-de-specialite-nsi, dans https://eduscol.education.fr/document/33178/download

Questions :

- nombre d'occurrences d'un caractère dans une chaîne
- rendu de monnaie (programme à trous)

Document : corrige-22-NSI-01.html

19 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 débuter. Ici on montre comment faire pour que le compilateur

- afficher les avertissements,

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

Document : configurer-codeblocks.html

20 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.

21 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 `Makefile`s qui vont avec.

Document : makefile-sous-projet.html

22 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

23 "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 $S \to \epsilon \| aSbS$.

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

24 "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

25 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 écrite 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.

26 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

27 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

28 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

29 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

30 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

31 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

32 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

33 Bases markdown

Le strict minimum, pour les débutants.

Document : bases-markdown.html