Mise à niveau Unix
TP 2

Juliusz Chroboczek

11 septembre 2015

Créez un répertoire (nommé par exemple `tp2`) dans lequel vous travaillerez durant ce TP. Avant de commencer, assurez-vous que votre fichier `.emacs` contient la ligne

```
(setq-default c-basic-offset 4)
```

Exercice 1 (Un premier programme C)**.**

1. À l'aide d'Emacs, créez un fichier `bonjour.c` dans lequel vous taperez le programme donné dans le premier poly de C. Commencez chaque ligne en appuyant sur la touche *tab*. À quoi sert la coloration syntaxique ?
2. Compilez votre programme à l'aide de la commande

   ```
   gcc -Wall bonjour.c
   ```

   S'il y a des erreurs, corrigez-les et recommencez. Quel fichier a été créé ? Exécutez-le à l'aide de la commande `./a.out`.
3. Supprimez le fichier `a.out`. Compilez votre programme de nouveau, mais cette fois-ci à l'aide de la commande

   ```
   gcc -Wall -o bonjour bonjour.c
   ```

   Quel fichier a été créé ? Exécutez-le.
4. Dans Emacs, tapez `M-x compile`. Remplacez la ligne de commande proposée par la ligne

   ```
   gcc -Wall bonjour.c
   ```

   puis validez.

Exercice 2 (Erreurs à la compilation)**.**

1. À l'aide du *shell*, copiez le fichier `bonjour.c` en un fichier `bonjour2.c`. Ouvrez-ce dernier à l'aide d'Emacs, et supprimez le point-virgule « `;` » de la ligne 8.
2. Compilez le programme `bonjour2.c` à partir du shell. Que se passe-t-il ?
3. Compilez le programme `bonjour2.c` à partir d'Emacs. Dans le tampon « `*Compilation*` », cliquez sur le message d'erreur à l'aide du bouton du milieu. Que se passe-t-il ?

Exercice 3. Écrivez un programme `nom.c` qui affiche votre nom. Compilez-le à l'aide de la commande « `gcc -Wall` », et testez-le.

Exercice 4. Écrivez un programme `agiste.c` qui demande à l'utilisateur son année de naissance puis affiche son âge. On supposera que ce programme ne sera utilisé qu'en 2015. Compilez et testez votre programme.

Exercice 5. Écrivez un programme `sexiste.c` qui demande à l'utilisateur le premier chiffre de son numéro de sécurité sociale puis répond « `Bonjour Madame.` » si c'est une femme (le chiffre vaut 2), et « `Bonjour Monsieur.` » sinon. Compilez et testez votre programme.

Exercice 6. Écrivez un programme `max.c` qui affiche le plus grand de deux nombres entiers entrés par l'utilisateur. Compilez et testez-le.

Exercice 7. Écrivez un programme `max3.c` qui affiche le plus grand de trois nombres entiers entrés par l'utilisateur. Compilez et testez-le.

Désormais, nous ne mentionnerons plus que *tous les programmes que vous écrirez doivent être compilés et testés*.

Exercice 8. Modifiez le programme `sexiste.c` pour qu'il affiche « `Bonjour Madame.` » si l'utilisateur est une femme (premier chiffre valant 2), « `Bonjour Monsieur.` » si c'est un homme (premier chiffre valant 1), et « `Bonjour alien.` » sinon.

Exercice 9. Écrivez un programme `un-deux.c` qui demande à l'utilisateur un entier positif puis affiche :

- « `aucun` » si ce nombre vaut 0 ;
- « `un` » si ce nombre vaut 1 ;
- « `deux` » si ce nombre vaut 2 ;
- « `plusieurs` » sinon.

Exercice 10. Écrivez un programme `bissextile.c` qui demande à l'utilisateur de rentrer le numéro d'une année puis indique s'il s'agit d'une année bissextile.

Exercice 11. Écrivez un programme `vitesse.c` qui demande le nombre de kilomètres parcourus et le temps de la balade en minutes et affiche la vitesse moyenne en kilomètres par heure. *(Attention : pourra-t-on travailler avec des entiers ?)*

Compilez et testez votre programme. Que se passe-t-il si l'utilisateur entre un temps de zéro ?

Exercice 12. Écrivez un programme `heure.c` qui demande à l'utilisateur l'heure de la fin du TP et l'heure qu'il est, puis affiche le nombre de minutes restant jusqu'à la fin du TP. On n'utilisera que des variables entières.