IN103 : Sujet du TP1

Groupe ESIEE, Denis BUREAU, février 2006, v2.

Attention ! Le sujet peut être modifié jusqu'à la veille du TP.

1 Les objectifs

Être capable de réaliser à partir d'un terminal sous linux des programmes en C.
Rappel : De nombreux liens sur C sont disponibles sur la page web de l'unité IN103.

2 Prise en mains

2.1 L'éditeur `nedit`

Taper `nedit hello.c &` pour lancer l'éditeur de texte en arrière-plan, puis cliquer sur `New file` pour créer le fichier `hello.c` qui contiendra le programme ci-dessous
(il est également possible de lancer `nedit` par le menu "KDE").

2.2 Premier programme

```
#include <stdio.h>
int main( void )
{
  printf( "Bonjour !\n" );
} /* main() */
```

2.3 La compilation

Pour utiliser la bonne version du C, il faut préciser au compilateur `gcc` quelques options.
Pour éviter d'avoir à les retaper à chaque compilation, `linux` nous permet de créer un *alias*.
Pour éviter d'avoir à recréer l'alias à chaque login, il faut le stocker dans le fichier `.cshrc` à la racine de votre compte, car ce fichier est automatiquement lu au début de chaque session.
Aide: taper `ls -a` pour voir TOUS les fichiers d'un répertoire.

Pour cela, lancer l'éditeur de texte sur ce fichier et insérer dans la partie Linux la ligne suivante :
`alias mcc  gcc -ansi -pedantic -W -Wall`
Fermer la fenêtre Terminal et en ouvrir une nouvelle pour prendre en compte ce nouvel alias. Revenir dans le bon répertoire de travail. Taper `alias` pour vérifier que `mcc` est maintenant disponible.

Pour lancer la phase de compilation, taper `mcc -c hello.c` ; un message d'erreur s'affiche ; ajouter l'instruction manquante puis relancer la commande de compilation jusqu'à ce qu'il n'y ait plus de message d'erreur ; le fichier `hello.o` doit être créé ; le vérifier.

Info : le pré-processeur est automatiquement lancé avant la compilation proprement dite et interprète toutes les lignes commençant par `#` .
En cas de besoin, la liste des options peut être obtenue par `man gcc` .

2.4 L'édition de liens

Pour lancer cette phase, taper `gcc -lm hello.o -o hello` ; si aucun message d'erreur ne s'affiche, le fichier `hello` doit être créé ; le vérifier.
Info : le lien avec la **l**ibrairie **m**athématique n'est utile que lorsque le programme utilise des fonctions déclarées dans `<math.h>` .

2.5 L'exécution

Pour lancer le programme, taper simplement `./hello` ; vérifier l'affichage.

2.6 Modification du programme

Modifier le programme pour qu'il pose la question `"numero ? "`, qu'il saisisse un entier (par exemple 6), puis qu'il affiche `"Bonjour numero 6 !"` . Sauvegarder le fichier et lire le paragraphe suivant.

2.7 Compilation & édition de liens enchaînées

Pour lancer les 2 phases automatiquement, taper `mcc -lm hello.c -o hello` ; si aucun message d'erreur ne s'affiche, le fichier `hello` doit avoir été modifié ; le vérifier.

Attention ! Ne pas lancer les 2 phases séparément est certainement une facilité lorsque tout se passe bien, mais peut être un handicap lorsqu'il y a des erreurs. Il faut alors déterminer si c'est une erreur de compilation ou bien une erreur d'édition de liens.

Par exemple dans l'exercice 3.4 ci-dessous, oublier la ligne `#include <math.h>` provoquera une erreur de compilation car la fonction `sqrt` sera inconnue du compilateur, mais oublier l'option `-lm` provoquera une erreur de l'éditeur de liens car il ne trouvera pas le code compilé de la fonction `sqrt`.
En général, tout message d'erreur ne commençant pas par `fichier.c:LL:CC` provient de l'éditeur de liens (LL = n° de ligne, CC = n° de colonne).

Pour chaque exercice ci-après (sections 3 & 4):

1. Créer un fichier `.c` différent pour chaque exercice ; il contiendra en plus du sous-programme demandé une fonction `main()` dont le schéma sera toujours le même :
   affichage d'un message, saisie de valeurs, appel de la fonction, affichage du résultat ;
   ou si c'est une procédure, affichage d'un message, saisie de valeurs, appel de la procédure.

2. Tester ensuite votre programme selon la procédure exposée sur la page web "Test des exercices", accessible à partir de la page web de l'unité à la fin du paragraphe TP1.

3 Première phase

1. Écrire en C une fonction `totalCm` qui prend en paramètre deux nombres entiers correspondant au nombre de mètres et au nombre de centimètres d'une longueur, et qui retourne le nombre total de centimètres.
   Exemple: paramètres : 3 , 72 → 372 cm
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

2. Écrire en C une fonction `periode` qui prend en paramètre un nombre réel correspondant à la longueur d'un pendule, et qui retourne la période correspondante. (rappel: $t = 2\pi\sqrt{\frac{L}{g}}$)
   Chercher `PI` dans le fichier `math.h` et définir `G` en constante.

   Attention ! Il semble y avoir un bug dans le compilateur *gcc* : pour pouvoir utiliser les constantes mathématiques, il faut faire précéder l'inclusion de `math.h` par la ligne `#define __USE_BSD`
   Exemple: paramètre : 2.4525 → 3.141592653589793
   Aide: format de saisie pour un double : `%lf`
   Aide: format d'affichage pour un double : `%.15g` ⇒ 15 chiffres après la virgule
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

3. Écrire en C une procédure `calc` qui simule une mini-calculatrice permettant les 4 opérations usuelles plus l'élévation à la puissance (`pow()`) . Elle prendra en paramètres d'entrée deux nombres réels, et un caractère représentant l'opération à effectuer entre ces deux nombres ; elle effectuera le calcul, puis affichera le résultat comme ci-dessous.
   Aide : `scanf(" %c",&op);` l'espace permet "d'avaler" un espace ou un retour à la ligne.
   Exemples :
   paramètres : `2` , `3.14` , `'*'` ⟶ affichage : `2 * 3.14 = 6.28`
   paramètres : `2` , `3` , `'^'` ⟶ affichage : `2 ^ 3 = 8`
   paramètres : `2` , `0` , `'/'` ⟶ affichage : `Division par zero !`
   paramètres : `2` , `3.14` , `'#'` ⟶ affichage : `Operation inconnue !`
   Cet exercice a déjà été réalisé en Java lors du TD5.1 d'IN101.

4. Écrire en C une fonction `factorielle` qui accepte les nombres négatifs :
   si $n < 0$, $n!$ vaudra - $(-n)!$.
   Rappels : $0! = 1$ et $n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$.

   Questions à se poser :
   Quel est le nombre maximum dont on peut calculer la factorielle avec cette fonction ?
   Pourquoi, simplement en voyant le résultat de 17!, sait-on immédiatement que le maximum est inférieur à 17 ? et pour 13! ?
   Pourquoi ce maximum est-il inférieur à celui de la version Java ?
   Exemples :
   paramètre : 5 → 120, paramètre : -6 → -720
   Cet exercice a déjà été réalisé en Java lors du TD5.2 d'IN101.

4 Deuxième phase

Cette phase peut être considérée comme du travail personnel. Elle commence pendant le TP dès que la première phase est terminée et ne s'achève que lorsque tous les exercices ont été réalisés et testés.
Les 7 premiers exercices sont des exercices d'entrainement, les 5 suivants sont plus conséquents.

1. Écrire en C une fonction `real` qui prend en paramètre deux nombres entiers correspondant au numérateur et au dénominateur d'un rationnel, et qui retourne la valeur réelle correspondante.
   Exemple: paramètres : 9 , 2 → 4.5
   Aide: format d'affichage pour un double : `%g`
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

2. Écrire en C une fonction `sommeJusquA` qui prend en paramètre un nombre entier N, et qui retourne la somme des N premiers entiers (sans faire de boucle !).
   Exemple: paramètre : 10 → 55
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

3. Écrire en C une fonction `surface` qui prend en paramètres trois nombres réels correspondant aux longueurs des trois côtés d'un triangle, et qui retourne sa surface.
   (rappel: $s = \sqrt{p\,(p-a)(p-b)(p-c)}$ où p est le demi-périmètre du triangle)
   Exemple: paramètres : 1.5 , 2. , 2.5 → 1.5
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

4. Écrire en C une fonction `estSur` qui prend en paramètres 5 réels correspondant d'une part aux coordonnées du centre et au diamètre d'un cercle, et d'autre part aux coordonnées d'un point. Cette fonction doit retourner 1 (vrai) si le point est sur le cercle, et 0 (faux) sinon (attention aux problèmes de précision. . .).
   Contrainte: définir la précision souhaitée en constante.
   Exemple: paramètres : 0.0 0.0 2.0 0.707107 0.707107 → 1 (à 10^{-6} près)
   Cet exercice a déjà été réalisé en Java lors du TD3 d'IN101.

5. Écrire en C une procédure `typeCarac` qui affiche le type du caractère passé en paramètre.
   Exemples : `'b'` ⟶ `minuscule`, `'B'` ⟶ `majuscule`, `'3'` ⟶ `chiffre`, `'+'` ⟶ `divers`.
   Cet exercice a déjà été réalisé en Java lors du TD5.1 d'IN101.

6. Écrire en C une fonction `max` qui retourne le plus grand des 3 entiers passés en paramètres.
   Contrainte : n'utiliser qu'un seul `return` dans cette fonction.
   Exemple : paramètres : `5` , `3` , `5` → `5`
   Cet exercice a déjà été réalisé en Java lors du TD5.1 d'IN101.

7. Écrire en C une fonction `voyelleMin` qui essaie de saisir une voyelle minuscule au maximum 3 fois.
   Cet exercice a déjà été réalisé en Java lors du TD5.2 d'IN101.

8. Écrire en C une fonction `isCPOS` déterminant si deux droites sont Confondues ou Parallèles ou Orthogonales ou Sécantes. La fonction prendra les 2×3 coefficients réels en paramètre et retournera un caractère `'c'`, `'p'`, `'o'`, ou `'s'`. Hiérarchiser les 4 cas, et ne pas oublier les cas particuliers tels que les droites horizontales ou verticales, ainsi que les coefficients qui ne définissent pas une équation de droite (on se contentera d'une précision de 10^{-6}).
   Vérifier son bon fonctionnement sur, notamment, chacun des exemples suivants :
   - les droites [1, 2, 3] et [3, 6, 9], qui sont confondues;
   - les droites [2, 0, 3] et [9, 0, 5], ou [0, 2, 3] et [0, 9, 5], qui sont parallèles;
   - les droites [2, 2, 0] et [1, -1, 1], ou [1, -1, 1] et [2, 2, 0], qui sont orthogonales;
   - les droites [1.23456789, 9.87654321, 2] et [9.8765432, -1.2345678, 3], qui sont orthogonales;
   - les droites [0, 1, 2] et [1, 2, 0], qui sont simplement sécantes;
   - [0, 0, 9] et [0, 0, 0], qui ne définissent pas des droites.
   Cet exercice a déjà été réalisé en Java lors du TD5.1 d'IN101.

9. Écrire en C une procédure `dernierDimanche` permettant, à partir d'une date quelconque (sous la forme jour/quantième/mois/année), de calculer et d'afficher la date du dernier dimanche du mois en question.
   Les paramètres d'entrée respecteront les règles syntaxiques suivantes :

```
<jour>      ::= 'l' | 'm' | 'M' | 'j' | 'v' | 's' | 'd'
<quantieme> ::= 1 | 2 | 3 | ... | 30 | 31
<mois>      ::= 1 | 2 | 3 | ... | 11 | 12
<annee>     ::= 1583 | 1584 | ... | 2999 | 3000
```

   On suppose les données fournies correctes et conformes aux spécifications.
   Exemple :
   paramètres : `j` , `9` , `12` , `1999` → affichage : `Le dernier dimanche du mois est le 26/12/1999`
   Contrainte : n'oubliez pas qu'il est interdit d'utiliser une boucle !
   Cet exercice a déjà été réalisé en Java lors du TD5.1 d'IN101.

10. Écrire en C la fonction `chiffresMots` qui affiche la suite de mots correspondant aux chiffres d'un nombre entier strictement inférieur à 1000000, passé en paramètre.
    Contraintes : ne pas effectuer de calcul de puissance et ne pas afficher de zéro en tête du nombre.
    Exemple : paramètre : 42307 → quatre deux trois zéro sept
    Cet exercice a déjà été réalisé en Java lors du TD5.2 d'IN101.

11. - Écrire en C une fonction auxilliaire `rom` qui prend comme paramètres un chiffre (1 ou 5) et une puissance de 10, et qui retourne le caractère romain correspondant.
Exemple : paramètres : 5, 100 → 'D'.
- Écrire en C (dans le même fichier que la fonction `rom`) une procédure `dec2rom` qui affiche le nombre en chiffres romains correspondant à l'entier compris entre 1 et 3999 passé en paramètre.
Contraintes : utiliser la fonction auxilliaire définie précédemment et ne pas recalculer la puissance à chaque fois.
Exemples :
2443 ⟶ `MMCDXLIII`, 1998 ⟶ `MCMXCVIII`, 7 ⟶ `VII`, 3215 ⟶ `MMMCCXV`,
2999 ⟶ `MMCMXCIX`, 1444 ⟶ `MCDXLIV`, 506 ⟶ `DVI`, 3888 ⟶ `MMMDCCCLXXXVIII`
Cet exercice a déjà été réalisé en Java lors du TD5.2 d'IN101.

12. Écrire en C une procédure `calMois()` permettant, à partir d'une date quelconque (mêmes paramètres qu'à l'exercice 9 ci-dessus), de calculer et d'afficher le calendrier du mois en question comme ci-dessous (M majuscule pour Mercredi).
Exemple : m, 6, 11, 2001 ⟶

```
Calendrier du mois de novembre 2001 :
   lundi    mardi Mercredi    jeudi vendredi   samedi dimanche
       .        .         .        1        2        3        4
       5        6         7        8        9       10       11
      12       13        14       15       16       17       18
      19       20        21       22       23       24       25
      26       27        28       29       30        .        .
```

Contrainte : tenir compte des années bissextiles.
Rappel :

- une année est bissextile si elle est divisible par 4
- exception : elle n'est pas bissextile si elle est divisible par 100
- exception de l'exception : elle est néanmoins bissextile si elle est divisible par 400

Cet exercice a déjà été réalisé en Java lors du TD5.2 d'IN101.