PIN203

T.P. numéro 1-Addenda
Introduction au C et à Unix

4 Découverte du language C

Le but de ce TD-TP est de se familiariser avec les premiers éléments de la syntaxe C à travers l'étude d'un petit programme. Un précieux soutien peut être trouvé pour les problèmes concernant le langage C. Les programmes que l'on va développer dans ces TP seront écrits dans un fichier texte grâce à l'éditeur de texte gedit (ou nedit ou emacs). Une fois le programme écrit, on le « compile » en utilisant un programme appelé « compilateur » (gcc dans notre cas) qui transforme le texte du programme en instructions directement exécutables par le processeur de la machine.

Exercice 1.7 : quelques règles de base

1. Sauvegarder le programme lineaire.c dans le répertoire TP1 déjà créé
2. Éditez-le avec gedit en lançant la commande gedit linaire.c depuis la ligne de commande d'une fenêtre terminal.

Afin de reconnaître votre listing sur imprimante et de donner une copie à l'enseignant : introduire en première ligne du programme lineaire.c un commentaire sur 2 lignes donnant, dans l'ordre(!):
vos noms (en CAPITALES), prénoms, groupe PIN203yy Atelier Info x

3. Que fait ce programme?
4. Faire un listing (ls) du répertoire TP1 : il ne doit y avoir que TP1.pdf et lineaire.c
5. Compilez le programme avec la commande gcc lineaire.c
6. Refaire un ls ; que constatez-vous ?
7. Executer le programme avec la commande : ./a.out

En fait cette méthode de compilation écrasera les fichiers exécutables précédemment compilés. Il est donc plus sur de donner un nom explicite à l'exécutable.

8. Effacer le fichier a.out (commande rm) et faire un ls : que se passe-t-il ?
9. Compilez maintenant lineaire.c avec la commande gcc lineaire.c –o lineaire puis refaire un ls.
10. Exécutez ce programme en tapant la commande: ./lineaire et constater que vous avez la même chose qu'en 7).
11. Pour s'assurer qu'aucune possibilité d'erreur de syntaxe n'existe dans le code celui-ci doit être compilé avec l'option –Wall (all warnings turned on) et donc vous devez maintenant compiler avec la commande : gcc lineaire.c –o lineaire –Wall

Désormais pour tous les TP vous compilerez obligatoirement avec la forme de commande précédente: gcc nomprog.c –o nomprog –Wall.
Désormais un programme ne sera jugé acceptable que s'il est compilé sans avertissement.

12. Détaillez les rôles des trois lignes finales de lineaire.c commençant par printf en commentant dans le programme même.

13. Exécuter le programme « lineaire ». Ce programme fait-il bien ce qu'il prétend faire ? Détaillez sur votre cahier de manip, l'algorithme proposé. Le programme peut-il traiter sans problème tous les jeux de valeurs a1, b1, c1, a2, b2, c2 ?

Exercice 1.8

L'objectif est de modifier le programme d'origine, lineaire.c, pour avoir plus de facilités et de sécurité d'exécution.

1. A l'aide de la commande cp faire une copie de lineaire.c en lineaire2.c. Modifier le programme lineaire2.c pour qu'il affiche à l'écran la preuve que la solution trouvée répond bien au problème. (par exemple en affichant a1*x+b1*y = c1 et a2*x+b2*y = c2 avec les valeurs des différentes variables)

2. Compléter le programme lineaire2.c pour entrer les 6 valeurs des paramètres a1, b1, c1, a2, b2, c2 de votre choix. Pour cela il faut :
- introduire une ligne « printf » demandant à l'utilisateur d'entrer les données en donnant le format ; c.a.d afficher à l'écran le message : « entrer a1,b1…… »
- introduire une ligne « scanf » pour que le programme lise les données entrées au clavier et les ranges dans les variables déclarées.

On aura vu la description du fonctionnement en faisant man scanf ou en relisant le manuel de cours.
On fera particulièrement attention au type des arguments de cette fonction.
On fera attention à se conformer au style standard du codage.
On vérifiera le bon fonctionnement du programme sur deux exemples bien choisis.

3. Le programme ne traite pas le cas particulier d'un dénominateur nul. Pour qu'il traite proprement ce cas on utilisera pour cela une instruction conditionnelle. On nommera lineaire3.c la nouvelle version de programme, pour la différentier dans le cahier de manip.
Introduire dans le programme le test de dénominateur non nul et faire les branchements aux instructions appropriées :

- Si dénominateur non nul : on continue les calculs et on sort du programme.
- Si dénominateur nul: afficher à l'écran un message « dénominateur nul », et sortir du programme sans faire les calculs.

4. Vérifier que votre programme marche correctement dans les 2 cas envisagés.

Remarques :
Il est rappelé que résultats et programmes sont à imprimer et coller soigneusement dans le cahier de manip, qui fera l'objet d'une évaluation dans le cadre de la note de contrôle continu. Chaque programme devra avoir été commenté dans son fichier source.
A la fin de séance une copie des programmes écrits par le groupe sera donnée à l'enseignant. Utiliser pour cela la commande : a2ps –Pimp0ati1 fichier.c pour l'imprimante de l'atelier1 (..ati2 pour atelier2, etc) ou imprimer directement à partir de l'éditeur de texte (gedit).

Note : Lors de l'exécution, on peut rediriger la sortie écran dans un fichier en utilisant l'opérateur de redirection > comme par l'exemple :

lineaire2 > resultat.dat

qui redirige la sortie écran vers le fichier *résultat.dat* (fichier ASCII) et permet d'archiver le résultat des calculs dans un répertoire et de l'imprimer avec la commande a2ps. Ce fichier doit aussi être archivé sur un répertoire non « cremi » accessible des « ateliers info libre service » !! essayer le webmail ……..pour l'envoyer au(à la) co-équipier(e).