POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5)
http://laure.gonnord.org/pro/teaching/

Programmation Structurée 2011/2012

TP2 Premiers programmes C sous emacs

Objectifs

Écrire, compiler, exécuter des programmes simples. Comprendre quelques erreurs de compilation. Utiliser `scanf` et `printf`.

1 Découverte Linux : mon terminal est intelligent

1. Le Terminal (la console) comporte une historique : avec flèche vers le haut, vous récupérez votre commande précédente, puis la précédente, ...
2. Le terminal *complète* les commandes. Essayez de taper `ge` puis TAB, il devrait compléter en `gedit` (si plusieurs commandes sont possibles, il vous les liste). Il complète aussi les chemins. Si vous désirez aller dans votre répertoire Algo et que vous tapez `cd Al` puis TAB, le chemin sera complété (si Algo existe à cet endroit, bien sûr).

Essayez !

2 Emacs

`Emacs` est un éditeur de texte très connu dans le monde Linux. Il permet d'éditer n'importe quel type de texte. Dans le cas de source C (et d'autres langages de programmation), l'éditeur réalise l'indentation automatique, et plein d'autres choses encore (on peut par exemple compiler dans emacs). La prise en main de ce logiciel peut paraître fastidieuse, mais avec de la pratique vous verrez toute sa puissance ! Dans ce TP, **l'utilisation d'Emacs est obligatoire** [1].

Syntax coloring sous emacs Récupérez sur la page web du cours le fichier `pointemacs`, mettez le dans votre racine (`~/`) et changez son nom pour qu'il devienne `.emacs` (oui, oui, un fichier qui commence par un point). Admirez la coloration syntaxique d'un fichier .c ouvert par `emacs` après cette manipulation (par exemple ouvrez le fichier `bonjour.c` du TP précédent).

Faites la même manipulation sur le compte des deux membres du binôme !

3 Programmation

Mise en place Avant de commencer la programmation, créez **à la ligne de commande** le répertoire `Algo/TP2`. Placez vous à l'intérieur de ce répertoire.

1. À la ligne de commande, lancez `emacs` en demandant la création et l'édition du fichier `tp2.c` :

1. Pour vous aider, on vous distribue la refcard emacs

`emacs tp2.c & RET`

2. Créez un programme C minimal qui réalise uniquement un `printf`. Sauvez (ne sortez pas d'emacs). Compilez, exécutez (ligne de commande, cf le cours et le tp1).

Programme, fonctions Vous ferez **toujours** précéder chaque fonction écrite d'un commentaire expliquant sa fonctionnalité. Par exemple pour `maxi3` :

```
//retourne le maximum des entiers a,b et c
int maxi3(int a, int b, int c){

  //code du maxi3
}
```

1. Dans le `main`, demandez trois entiers à l'utilisateur (cf cours 2), puis imprimez leur somme et leur produit. Compilez, testez.
2. Écrire une fonction `int maxi3(int a, int b, int c)` (avant le main) qui calcule le maximum de 3 entiers (un code similaire se trouve dans le cours).
3. Dans le `main`, faire un appel à cette fonction pour calculer le maximum de 3,7,-20, puis le maximum de 3,2,x où x a été demandé à l'utilisateur. Compilez, testez.
4. Écrire une fonction `bool ordrecroissant(int x,int y, int z)` qui répond `true` si $x \leq y \leq z$ et `false` sinon. Compilez.
5. Appelez cette fonction dans le `main` sur les triplets $(2, 3, 4)$, $(5, 1, 60)$ et $(t, u, 70)$ avec t, u demandés à l'utilisateur. À chaque fois, récupérez le résultat de la fonction, et imprimer "oui !" si le résultat est croissant, "non !" sinon. Compilez, testez.
6. Sur le même modèle, écrivez des fonctions qui prennent trois arguments entiers et qui répondent aux questions suivantes :
   - Sont-ils tous égaux ?
   - Sont-ils distincts deux à deux ?
   - L'un des trois est-il plus grand que la somme des autres ?

▶Ces fonctions seront utilisées dans le TP3.