

TP N° 1

Outils d'aide à la programmation

But du TP :

Découvrir les outils séparés d'aide à la programmation en utilisant la ligne de commande pour :

1. connaître les parties cachées du développement par les IDE.
2. apprécier par la suite l'apport de simplification des IDE sans être victime de leur effet boite noire.

Énoncé :

Soit l'algorithme écrit sur papier :

```
Algorithme simple ;  
    var x,y,som : entier ;  
debut  
    lire(x,y) ; {j'ai omis le message ici de façon volontaire}  
    som ← x+y ;  
    ecrire(som) ; {idem}  
fin.
```

On se propose de traduire l'algorithme en programme et de l'écrire en utilisant un éditeur de textes. Ensuite, on utilisera la ligne de commandes pour le compiler puis pour l'exécuter. Voici une proposition du code C associé :

```
#include <stdio.h>  
  
int main()  
{  
    int x,y,som;  
    scanf("%d %d",&x,&y);  
    som=x+y;  
    printf("%d",som);  
    return 0;  
}
```

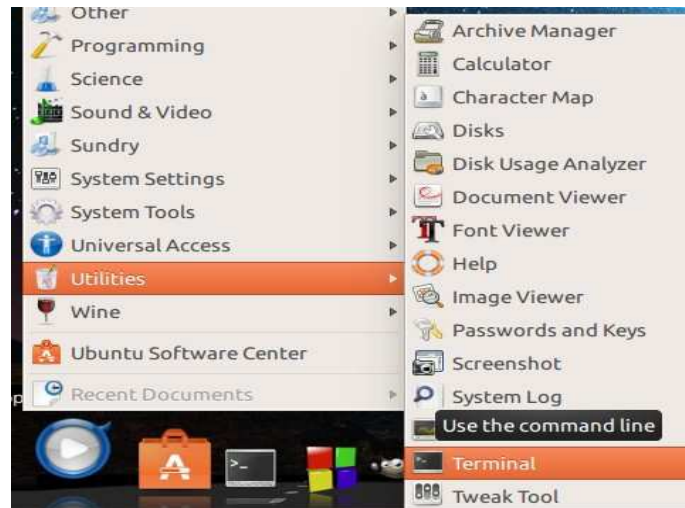
- Que fait scanf dans ce code? printf ? Où est l'affectation ?
- Le code algorithmique suivant réalise-t-il la même chose ? Why ? Réalise-t-il un calcul ?

Algorithme simple2 ;

```
    var x,y,som : entier ;  
debut  
    ecrire('donner les valeurs de x et y') ;  
    som←x+y ;  
    ecrire('Le resultat est :som') ; {oui som est dans le message}  
fin.
```

Sous Linux :

Ouvrir un terminal. Il s'agit d'une fenêtre dans laquelle on peut taper des commandes systèmes.



On obtient une fenêtre nous invitant à entrer une commande :

```
home@votreOrdinateur:~$ _
```

A. Quelques commandes à connaître :

1. `ls`: permet de lister le contenu d'un dossier. Tapez là dans le terminal :

```
home@votreOrdinateur:~$ ls
```


si vous voulez plus de détails sur le contenu de notre répertoire, on tape `ls -all`
2. `mkdir`: permet de créer un répertoire. On peut par exemple créer un répertoire pour nos tp:

```
home@votreOrdinateur:~$ mkdir tpc
```
3. `cd` : changer de répertoire : on l'utilise pour nous déplacer vers notre répertoire :

```
home@votreOrdinateur:~$ cd tpc
```

B. Editer le code source avec gedit:

Lancer l'éditeur gedit par la commande : (& pour pouvoir utiliser le terminal en parallèle)

```
home@votreOrdinateur:~/tpc$ gedit &
```

Puis Taper le programme traduisant l'algorithme ci-dessus dans l'éditeur.

L'enregistrer avec le nom `prgSimple.c` dans le dossier `tpc`.

Vérifier que le fichier est bien au bon endroit par `ls`.

C. Compilation avec gcc:

On peut maintenant le compiler par `gcc`

```
home@votreOrdinateur:~/tpc$ gcc -c prgSimple.c
```

Vérifier que le fichier `prgSimple.o` a été créé. Si ce n'est pas le cas corrigez les erreurs puis recompiler.

D. Edition de lien par gcc:

Taper la commande :

```
home@votreOrdinateur:~/tpc$ gcc -Wall prgSimple.c -o prgSimple
```

On obtient le fichier exécutable `prgSimple` (vérifier avec `ls`) que l'on peut lancer par `./prgSimple`

Comparer la taille du code objet avant et après édition de lien puis commenter.

Sous Windows :

Lancer la commande cmd (ou command) qui ouvre une fenêtre dans laquelle on peut taper des commandes systèmes.



On obtient une fenêtre nous invitant à entrer une commande :

```
C:\Users\Public>_
```

A. Quelques commandes à connaître :

1. dir: permet de lister le contenu d'un dossier. Tapez là dans le terminal :

```
C:\Users\Public> dir
```
2. md: permet de créer un répertoire. On peut par exemple créer un répertoire pour nos tp:

```
C:\Users\Public> md tpc
```
3. cd : changer de répertoire : on l'utilise pour nous déplacer vers notre répertoire :

```
C:\Users\Public> cd tpc
```

B. Editer le code source avec gedit:

Lancer l'éditeur Bloc-notes par la commande notepad :

```
C:\Users\Public\tpc> notepad
```

Puis Taper le programme traduisant l'algorithme ci-dessus dans l'éditeur.

L'enregistrer avec le nom prgSimple.c dans le dossier tpc.

Vérifier que le fichier est bien au bon endroit par dir.

C. Compilation avec gcc:

On peut maintenant le compiler par gcc

```
C:\Users\Public\tpc> gcc -c prgSimple.c
```

Vérifier que le fichier prgSimple.o a été créé. Si ce n'est pas le cas corrigez les erreurs (le chemin du compilateur doit être connu par le système. On peut utiliser pour cela la commande (avec le bon chemin) `PATH=c:\Program files\codeblocks\MinGW\bin;%PATH%` puis recompiler.

D. Edition de lien par gcc:

Taper la commande :

C:\Users\Public\tpc> gcc prgSimple.c -o prgSimple

On obtient le fichier exécutable prgSimple.exe (verifier avec dir) que l'on peut lancer par prgSimple

Comparer la taille du code objet avant et après édition de lien puis commenter.

Sous Windows ou Linux :

E. Importance des messages :

Reprendre depuis l'étape B. mais cette fois-ci en modifiant le code source par l'ajout de messages et des saut de ligne. Commenter.

F. Utilisation de commentaires :

En utilisant // ou /* et */ ajouter avec votre éditeur de texte des commentaires pour mettre en valeur les parties acquisition de données, traitement, et affichage de résultat.

Vérifier que l'ajout de commentaire n'a pas d'influence sur l'exécution de votre programme.

G. Pour s'assurer que votre machine n'est pas intelligente :

Modifier le code en mettant produit à la place de som comme suit :

```
#include <stdio.h>

int main()
{
    int x,y,produit;
    scanf("%d %d",&x,&y);
    produit=x+y;
    printf("%d",produit);
    return 0;
}
```

- En entrant 10 et 20 le résultat serait il 200 ou 30 ?
- Pourquoi l'ordinateur n'arrive pas à calculer le "bon" résultat ?
- Les identifiants utilisés sont-ils compris par la machine ? Donc pourquoi on doit utiliser des identifiants significatifs (et pour qui) ?

H. Introduction aux erreurs et mises en garde (warning) :

- Supprimer x de la partie déclaration, sauvegarder puis recompiler. Commenter. Essayer de l'exécuter.
- Restituer le code à son état initial. Ajouter & avant la variable à afficher, sauvegarder puis recompiler. Essayer de l'exécuter. Commenter.