Compilation des programmes en language C

>>> sous Linux (distribution Debian)

Description :

Parmi tous les systèmes d'exploitation, Linux est connu pour sa souplesse lors de la mise en place d'environnement de développement. Ici, il s'agira d'utiliser deux utilitaires permettant d'écrire et de compiler des programmes en langage C : Vim et GCC.

Compilation des programmes en language C

>>> sous Linux (distribution Debian)

Sommaire :

I) Préparation de l'environnement de programmation
1) Vim : l'éditeur de texte
2) GCC : le compilateur
II) Écriture, compilation et exécution du premier programme
1) Écriture du programme
2) Compilation du programme
3) Exécution du programme

Conclusion

I) Préparation de l'environnement de programmation

1)Vim : l'éditeur de texte

Vim est un éditeur de texte en ligne de commande. Son principal intérêt est de pouvoir colorer les lignes de code en fonction du langage utilisé.

— Installation de Vim :
Dans la plupart des distributions, Vim n'est pas installé. Utilisez la commande suivante pour l'installer dans votre environnement :

```
# aptitude install vim
```

Une fois installé, ouvrir le fichier de configuration de Vim à l'aide de la commande suivante :

```
# vim /etc/vim/vimrc
```

— Configuration de votre éditeur :
Parcourez le fichier à la recherche de la ligne 'synthax on. Le caractère " ' " permet de faire un commentaire, dé-commentez donc cette ligne. Pour enregistrer et quitter un fichier édité avec Vim appuyez sur [echap] et saisissez *:wq*

Voilà, votre éditeur est en place, on peut donc passer au compilateur.

Pour plus d'informations sur Vim est ses secrets ... c'est par ici !

2) GCC : le compilateur

Même plus besoin de vous le présenter, tellement **GCC** est connu sur la toile. Il s'agit d'un compilateur très puissant, permettant de compiler des programmes écrit en langage C et, aussi, en C++.

— installation de GCC :
Ce paquet s'installe très simplement à l'aide de la commande suivante :

```
# aptitude install gcc
```

Ensuite, et bien ... votre environnement de développement est prêt !

II) Écriture, compilation et exécution

1) Écriture du premier programme

Ouvrez une fenêtre "*console*", et saisissez :

```
# vim test2.c
```

Nota : pour le nom du fichier, mettez ce que vous voulez, d'ailleurs si je voulais je l'aurais appelé tartempion.c ... :P

Ensuite dans ce fichier, saisissez les lignes de code suivantes :

```
#include <stdio.h>                                          // Déclaration de la
bibliothèque utilisée dans l'exemple

char car;                                                   // Déclaration d'une
variable de type caractère

void main (void)                                          // Fonction principale
{
    printf("Salut ! ça va ?   [o/n]\n");                  // Fonction affichage
d'une chaîne de caractère
    scanf("%c", &car);                                      // Récupération du
caractère par une lecture du clavier

    if(car == 'o')          // Traitement de la première réponse
    {
        printf("OK ! cool passe une bonne journée alors ! :D \n\n");
    }
    else if (car == 'n')     // Traitement de la deuxième réponse
    {
        printf("Booo ! Allez la journée va bien se passer ! By!\n\n")
    }
    else
    {
        printf("OH ! Le gros naze ! Tu ne sais même pas te servir d'un clavier !
```

```
honte à toi ! \n");
        printf("[Computer out]\n\n");
    }
}
```

Quittez en sauvegardant (rappel : [echap] puis :wq).
Maintenant, il ne vous reste plus qu'à passer à la compilation proprement dite ! Pour cela, là encore, rien de plus simple, utiliser la commande suivante :

2) Compilation du premier programme

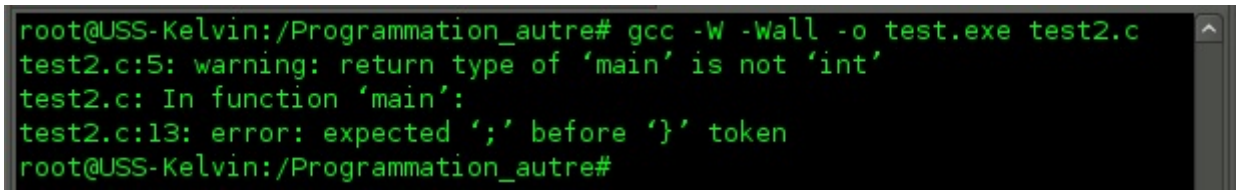
Pour cela, utiliser la commande suivante :

```
# gcc -W -Wall -o test2.exe test2.c
```

Nota : *Cette commande est exactement la même dans un environnement BSD (testé sous FreeBSD).*

— Explication de la commande : On peut distinguer plusieurs paramètres :
– -W permet d'indiquer le format du fichier compilé (ici il s'agit d'un fichier exécutable de type ELF, équivalent du ".exe" sous Linux).
– -Wall permet d'afficher tous les "warning" à l'utilisateur.
– -o *file_name* permet de spécifier le nom du fichier compilé.
– file_name.c est le nom du fichier source en langage C à compiler.

Si il y a des erreurs ou des "warning" dans votre code, **GCC** vous les retourne sous cette forme :



— Explication du warning : la seconde ligne dans la *console* annonce : **test2.c : 5 : Warning : return type of 'main' is not 'int'**. Ce Warning indique qu'en règle générale, votre fonction devrait retourner un entier mais que cette condition n'est pas forcément nécessaire au bon fonctionnement de votre programme (d'où le simple niveau de "Warning" : attention !). Pour ne plus le voir, il suffit de remplacer (dans le fichier test2.c) la ligne **void maint (void)** par **int main(void)** et d'ajouter **return 0 ;** juste avant la fermeture de la fonction main().

— Explication de l'erreur repérée par GCC : l'avant-dernière ligne de l'image annonce : **test2.c:13 : error : expected ' ;' before '}' token**. En fait gcc vous dit simplement que vous avez oublié un point-virgule avant le caractère '}'. GCC indique aussi que ce caractère ce trouve à la ligne 13 et que, par conséquent, l'erreur se trouve donc à la ligne 12.

Attention : cette indication n'est pas toujours vrai, tout dépend de l'erreur, dans certains cas GCC donne la ligne où se trouve l'erreur.

Il ne vous reste plus qu'à rééditer le fichier test2.c, allez à la ligne 12 ajouter le ' ;' , enregistrez et re-compilez le. Une fois qu'il n'y a plus d'erreurs ni de "warning", vous pouvez passer à l'exécution du programme.

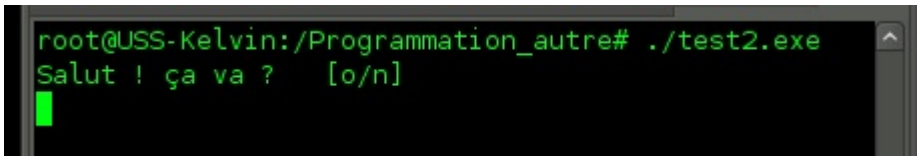
3) Exécution du premier programme compilé

Pour observer le fonctionnement de votre programme, il suffit d'utiliser la commande :

(Sous DEBIAN)

```
# ./test2.exe
```

et la console vous affiche la première phrase et attend votre réponse, comme sur l'impression d'écran suivante :



Conclusion

Voilà ! Vous avez vu comment mettre en place facilement un environnement de programmation simple et assez basique sous un environnement Linux.