Compilation séparée

Il devient très vite intéressant de séparer le contenu
d'un programme en plusieurs fichiers, car :

- tout recompiler prend du temps, inutilement
- la maintenance ne sera pas la même pour toutes les parties du programme
- c'est plus logique pour la compréhension :

un nom de fichier judicieusement choisi apporte de l'information
(inversement pour un nom mal choisi)

- ce découpage contribue à la maintenabilité et a la lisibilité du contenu

Exemple :

afficher.c
main.c
compter.c
entree_donnees.c
stockage.c
aide.c

Associés aux fichiers d'en-tête :

afficher.h
compter.h
entree_donnees.h
stockage.h
aide.h

+ ceux de la libC utilisés : stdio.h, stdlib.h ...etc

Problème : soit une fonction bidon() , utilisée dans plusieurs fichiers en .c

On suppose que sa definition se trouve dans defs.h

Comment réaliser l'inclusion de la définition de cette fonction dans plusieurs fichiers, afin que le compilateur ne fasse pas une inclusion multiple de defs.h , c'est à dire définisse plusieurs fois cette fonction (ce qui pose problème, provoque au mieux des warnings, au pire l'arrêt de la compilation en C++)

Solution proposée :

déclarer une seule fois la définition de la fonction dans defs.h, avec le mot clé " **extern** ".

Contenu de defs.h

```
....
#ifndef DEFS_H
#define DEFS_H

extern
void bidon()
{
        .....
}

#endif  /* DEFS_H  */
```

Mécanisme :

Dans chaque fichier d'implémentation (en .c)
dans lequel bidon() sera utilisée, juste ajouter la ligne

#include "defs.h"

Si le fichier d'en-tête se trouve dans le répertoire contenant
les sources lors de la compilation, sinon,
déclarer un repertoire include au compilateur (- Ipath_to_the_dot_h) dans le makefile.

Méthode générale :

Le fichier d'en-tête sources commence par les lignes :

```
#ifndef  m
#define m

....

/* et se termine par  :  */

#endif  /* m  */
```

m est un nom construit à partir du nom de fichier et
des caractères **underscore** pour rappeler l'extension .h

Remarque : à l'opposé d'une fonction externe, le mot clé **static** placé dans l'en tête d'une fonction rend sa déclaration locale au fichier. C'est une bonne habitude à prendre que d'essayer de limiter la portée d'une fonction, car cela évite les efffets de bords (il n'est pas bon que tout soit visible et accessible depuis n'importe quelle partie du programme).