

PoP Série 0 Solution

H1 : La commande `make` et la compilation séparée

H2 : La solution est dans un document séparé

Exercice 1 (niveau 0) : Notions de module (interface et implémentation)

Si on veut appeler la fonction `g()` (définie dans `B.cc`) dans la fonction `h()` (définie dans `A.cc`) alors il suffit que le prototype de `g()` soit visible dans `B.h` et que ce fichier en-tête soit inclus dans `A.cc`.

En effet le compilateur `g++` pourra vérifier que l'appel est correct lorsqu'il compilera `A.cc` (avec l'option `-c`) et cela permettra de produire `A.o`. Il n'est pas nécessaire d'en connaître plus de la fonction `g()` lors de la compilation de `A.cc`.

Exercice 2 (niveau 0) : Le graphe des dépendances des fichiers

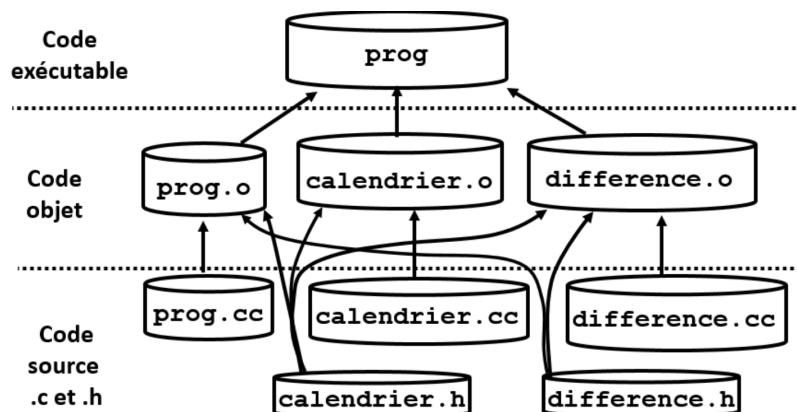


FIGURE 1 – Graphe des dépendances

Exercice 8 (niveau 1) :

- Fermez la fenêtre d'édition du `Makefile`, si elle était ouverte.
- Tapez `make clean`.
- Tapez `make depend`.
- Ouvrez, avec `geany`, le fichier source `prog.cc`.
- Lancez la commande `make` depuis `geany` (voir exercice 6) ou tapez simplement `make` dans le terminal. Observez les commandes et messages qui s'affichent dans la sous-fenêtre de `geany`, et qui montrent la progression, étape par étape. Combien d'appels à la commande `g++` sont effectués ? Pour chaque appel, indiquez s'il s'agit d'une compilation ou d'une édition de liens.

Il y a quatre appels à `g++`, trois pour compiler et un pour éditer les liens.

- Exécutez le programme (`prog`) dans le terminal et vérifiez brièvement qu'il fonctionne.

- Relancez la commande `make`. Que se passe-t-il cette fois ? Pourquoi ?

Aucune compilation ni édition des liens n'est effectuée, car `prog` est à jour (il n'y a eu aucune modification depuis le dernier appel à `make`).

- Modifiez `prog.cc` (par ex. le message d'introduction), sauvegardez. Relancez `make`. Combien de compilations sont faites ? Pourquoi ? Exécutez le programme.

Une seule compilation (re-génération de `prog.o`) à laquelle s'ajoute une édition de liens pour mettre à jour l'exécutable. Raison : voir le graphe des dépendances.

- Modifiez `difference.h` (par ex. un commentaire), sauvegardez. Relancez `make`. Combien de compilations sont faites ? Pourquoi ?

Deux compilations (re-génération de `prog.o` et `difference.o`) auxquelles s'ajoute une édition de liens pour mettre à jour l'exécutable. Raison : voir le graphe des dépendances.

- Modifiez `calendrier.h` (par ex. un commentaire), sauvegardez. Relancez `make`. Combien de compilations sont faites ? Pourquoi ?

Trois compilations (re-génération de `calendrier.o`, `difference.o` et `prog.o`) auxquelles s'ajoute une édition de liens pour mettre à jour l'exécutable. Raison : voir le graphe des dépendances.

- Modifiez `calendrier.cc` (par ex. un commentaire), sauvegardez. Relancez `make`. Combien de compilations sont faites ? Pourquoi ?

Une seule compilation (re-génération de `calendrier.o`) à laquelle s'ajoute une édition de liens pour mettre à jour l'exécutable. Raison : voir le graphe des dépendances.

- Effacez le fichier exécutable `prog` depuis une fenêtre Terminal, et relancez `make`. Combien de compilations sont faites ? Pourquoi ?

Aucune, car les modules objet sont à jour. `g++` est seulement appelé pour l'édition de liens pour mettre à jour l'exécutable.

- Faites `make clean`, puis `make`. Combien de compilations sont faites ? Pourquoi ?

Trois compilations sont effectuées, car les modules objet doivent tous être re-générés. S'y ajoute une édition de liens pour mettre à jour l'exécutable.

Ajoutez un module `siecle` dans le projet : créez les fichiers `siecle.h` et `siecle.cc`, et écrivez une fonction (prototype : `int siecle (int annee);`) qui retourne le siècle de l'année passée en argument (par exemple : 20, pour l'année 1971). Appelez la fonction `siecle( )` depuis le programme principal (`main`), pour afficher le siècle des deux dates fournies par l'utilisateur (par exemple : "Cette date se trouve dans le 20ème siècle"). Ensuite, mettez à jour le `Makefile` pour qu'il tienne compte du nouveau module. Enfin, générez un exécutable et testez votre programme. N'avez-vous rien oublié ?

Il ne faut pas oublier d'ajouter `siecle.cc` dans la macro `CXXFILES` du `Makefile`, d'inclure le fichier `siecle.h` dans le fichier source `prog.cc`, et de faire un `make depend`.