TP4 : Compilation séparée, Makefile

CSH : Initiation au C et au shell

Première année

[TP4]

1 Compilation séparée, Makefile

★ **Exercice 1. Compilation séparée**

- Vérifiez que les commandes `gcc -c code1.c` et `gcc -c code2.c` ne produisent pas de message d'erreur et que les fichiers ***code1.o*** et ***code2.o*** ont été créés.
- À partir de ces deux fichiers et de ***codage.c***, créez le fichier exécutable ***codage*** par la commande `gcc -o codage codage.c code1.o code2,o`
- Exécutez-le, puis supprimez les fichiers ***code1.o***, ***code2.o*** et ***codage***.

★ **Exercice 2. Utilisation du Makefile**

Lisez attentivement le contenu du fichier ***Makefile***. Notez bien que les lignes de *commande* commencent par un caractère de tabulation, et **non pas par des espaces**.

▷ **Question 1.** Exécutez la commande `make`. Quels fichiers ont été créés ?

▷ **Question 2.** Exécutez à nouveau la commande `make`. Que se passe-t-il ?

▷ **Question 3.** Supprimez le fichier ***codage*** et exécutez de nouveau la commande `make`. Que se passe-t-il ?

▷ **Question 4.** Éditez le fichier ***code1.c*** et modifiez-le en supprimant le premier caractère, puis en le remettant. Exécutez à nouveau la commande `make`. Que se passe-t-il ?

▷ **Question 5.** Changez le nom de ***codage.c*** en ***cod.c***, et exécutez la commande `make`. Que signifie le message d'erreur ?

▷ **Question 6.** Notez que ce message commence par le nom de la commande (ici, `make`) : quels autres messages avez-vous déjà rencontrés qui commence par le nom de la commande suivi du caractère ' :' ?

Éditez le fichier ***Makefile***, ajoutez à la fin les deux lignes suivantes : `codage.c : cod.c` et `<TAB>cp cod.c codage.c`. Exécutez la commande `make`. Il n'y a plus d'erreur.

▷ **Question 7.** Expliquez le rapport entre les deux lignes ajoutées au fichier ***Makefile*** et la disparition de l'erreur précédente.

★ **Exercice 3. Écriture d'un Makefile**

Vérifiez que vous pouvez créer un fichier exécutable ***instant_suivant*** en compilant ***instant_suivant.c***. Modifiez ***instant_suivant.c*** en ajoutant la ligne

```
attendre(1) ;
```

juste avant la ligne

```
printf("Nouvelle heure ......
```

Vérifiez que l'on peut alors créer le fichier "objet" ***instant_suivant.o***, mais pas le fichier exécutable : c'est-à-dire que la commande suivante "marche" : `gcc -c instant_suivant.c`

mais que celle-ci "ne marche pas" : `gcc -o instant_suivant instant_suivant.c`

▷ **Question 8.** Que comprenez-vous au message d'erreur obtenu avec cette dernière commande ?

Remarquez la ligne commençant par `ld:` ; l'erreur se produit à l'exécution d'une commande qui s'appelle `ld`.

Supprimez le fichier ***instant_suivant.o***.

En vous inspirant de ce qu'on vous a dit ci-dessus pour créer le fichier exécutable ***codage***, créez un fichier exécutable ***instant_suivant*** à partir de ***instant_suivant.c*** et de ***delai.o***, et exécutez le code obtenu.

Effacez les fichiers ***instant_suivant*** et ***delai.o***. Vérifiez qu'on peut compiler ensemble des fichiers qui ne sont pas dans le même répertoire : compilez ***instant_suivant.c*** à l'aide du fichier ***delai.o*** du répertoire ***/home/depot/1A/CSH/TP4*** en exécutant la commande suivante, puis exécutez le code obtenu. `gcc -o instant_suivant instant_suivant.c /home/depot/1A/CSH/TP4/delai.o`

Sauvegardez le fichier ***Makefile*** sous le nom ***Makefile_codage***. La commande `make` utilise systématiquement le fichier de nom ***Makefile*** : si l'on veut tester plusieurs *Makefiles*, il faut donc les sauvegarder, et attribuer le nom ***Makefile*** au fichier "courant".

Écrivez un fichier ***Makefile*** afin que la commande `make` crée ***instant_suivant*** à partir des fichiers ***instant_suivant.c*** et ***/home/depot/1A/CSH/TP4/delai.o***.

▷ **Question 9.** Il s'agit ensuite de tester ce ***Makefile***. Quelles commandes exécutez-vous pour cela ?

Remarque. La commande `touch`

Cette commande est utile pour effectuer des tests : sans changer le contenu d'un fichier, elle permet de mettre à jour sa date de dernière modification. Elle permet ainsi de simuler la modification d'un fichier texte sans passer par un éditeur. Elle permet aussi de créer un fichier vide.

Expérimentez les commandes suivantes : `ls -l` (Notez la date de ***instant_suivant.c***)

`make` `touch instant_suivant.c` `ls -l` `make` `touch insistant_suivant.c` `ls -l`

Supprimez le fichier ***instant_suivant***. Créez le fichier "objet" ***instant_suivant.o***. Créez le fichier exécutable ***instant_suivant*** par la commande :

```
gcc -o instant_suivant instant_suivant.o /home/depot/1A/CSH/TP4/delai.o
```

Exécutez ***instant_suivant***, puis effacez-le. Modifiez le ***Makefile*** pour que ***instant_suivant*** soit créé par la commande `make` à partir de ***instant_suivant.o***. Testez ce ***Makefile***.

Exécutez `touch instant_suivant.c`, puis `make`. Si vous obtenez le message `'instant_suivant' is up to date` ou un message d'erreur, c'est que votre ***Makefile*** est incorrect, alors corrigez-le.

2 Un peu de shell...

Lisez le programme ***jeboucle.sh***, et exécutez-le (vous devez l'arrêter par $<CTRL\text{-}C>$).
Inspirez-vous de ce programme pour écrire un programme ***date.sh*** qui exécute la commande `date` toutes les 5 secondes, testez-le.

▷ **Question 10.** Quel est le contenu de votre programme ?

3 Enchaînement des exécutions

▷ **Question 11.** Lisez le texte de ***date_vers_heure.c***, et compilez-le. Comment testez-vous son exécution ?

▷ **Question 12.** Modifiez ***date_vers_heure.c*** afin qu'il s'exécute sans fin en lisant une date et en affichant l'heure. Comment faites-vous pour utiliser les sorties de ***date.sh*** comme entrées de ***date_vers_heure*** ?

4 Erreurs et ennuis...

Lisez, compilez, et exécutez le programme ***un_a_dix.c***. Que fait ce programme ? Ce n'est visiblement pas ce qui était voulu par le programmeur.

▷ **Question 13.** Où est l'erreur ? Corrigez, compilez et exécutez.

Dans le fichier ***Makefile***, remplacez les caractères $<TAB>$ qui sont au début des lignes de commande par des espaces, puis exécutez la commande `make` (après avoir utilisé la commande `touch` au préalable).

▷ **Question 14.** Que comprenez-vous au message d'erreur ?

5 Exercices pour la prochaine fois

▷ **Question 15.** Ecrivez un ***Makefile*** permettant de produire l'exécutable ***exo*** à partir des fichiers ***exo.c*** et ***lire.c*** suivants en utilisant la compilation séparée.

exo.c

```
int main() {
  int i, x ;
  for (i=1 ; i<=10 ; i++) {
     x = lire_entier() ;
     printf("vous avez saisi l'entier %d\n", x);
  }
  return 0 ;
}
```

lire.c

```
#include <stdio.h>

int lire_entier() {
    int x ;
    printf("Un entier svp !\n");
    scanf("%d", &x) ;
    return x ;
}
```

▷ **Question 16.** Même question si l'on choisit de remplacer dans ***exo.c*** l'instruction `printf(...)` par l'appel d'une fonction `ecrire_entier` fournie dans un fichier ***ecrire.c***.

▷ **Question 17.** On suppose que l'on a exécuté avec succès la commande `make`, puis que l'on modifie ***ecrire.c***. Quelles commandes de compilation seront effectuées si l'on exécute de nouveau `make` ?