Projet Programmation 1

Compilation et Makefile

Amélie LEDEIN Louis LEMONNIER Yoan GERAN

Aujourd'hui

La compilation de fichier OCaml se fait à l'aide de la commande `ocamlc`, elle prend en paramètre le nom du fichier à compiler et produit un exécutable du nom `a.out`. De plus, l'option `-o` permet de spécifier le nom de l'exécutable à générer.

Exercice Télécharger l'archive `compile_one.tar.gz`, la décompresser, compiler le fichier pour obtenir un exécutable `prog`, puis exécuter le programe obtenu.

1 Avec plusieurs fichiers

Dans un gros projet, il est très rare d'avoir un seul fichier de code. En effet, le code sera plutôt organisé dans différents modules, et le fichier `foo.ml` aura possiblement besoin de fonctions écrites dans `bar.ml`.

Cependant le fichier `foo.ml` ne peut pas utiliser toutes les fonctions de `bar.ml`, il ne peut utiliser que celles déclarées dans un fichier dit d'**interface**, dont l'extension est `.mli`.

La compilation se fait alors en donnant à `ocamlc` les noms de tous les fichiers nécessaires à compiler.

Exercice Télécharger l'archive `compile_many.tar.gz`, la décompresser et observer les fichiers. Les compiler avec la commande suivante.

```
ocamlc name.mli name.ml main.ml -o main
```

Le fichier `name.ml` contient une fonction `check_name` non déclarée dans `name.mli`. Cette fonction ne pourra donc pas être utilisée dans le fichier `main.ml`.

Notons de plus que les fichiers doivent être donné dans le « bon » ordre à `ocamlc`. En effet, `main.ml` doit être compilé après `name.ml` puisqu'il en dépend.

Plutôt que de compiler à chaque fois chaque fichier `ml`, même ceux qui n'ont pas été modifiés, on peut se contenter de ne recompiler que ceux qui ont été modifiés. En fait, la compilation d'un fichier `.ml` produit un fichier `.cmo` et celle d'un fichier `.mli` produit un fichier `.cmi`. Et on peut alors les utiliser pour la compilation. Finalement, la compilation se fera plutôt ainsi.

```
ocamlc -c name.mli # produit un fichier name.cmi
ocamlc -c name.ml # produit un fichier name.cmo
ocamlc name.cmo main.ml -o main
```

2 Makefile

Finalement, dans un projet contenant plusieurs fichiers, il est plus simple de passer par un outil extérieur pour gérer la compilation. Généralement, il s'agit d'un Makefile.

Un fichier Makefile se compose de cette manière.

```
all: cible1 cible2
        commande_all

cible1: dependance1
        commande_cible1

cible2: dependance2
        commande_cible2
...
```

Cela signifie que pour faire `all`, on a d'abord besoin de `cible1` et de `cible2` et qu'ensuite on exécutera la commande `commande_all`. Les autres lignes se comprennent de la même manière.

En lançant la commande `make <cible>`, on demande alors à faire `<cible>`; si des dépendances sont manquantes, les commandes pour les avoir seront exécutées. De plus, avec `make`, on exécute la cible `all` (qui correspond généralement à la génération du programme entier).

Exercice Télécharger l'archive `compile_make.tar.gz`, la décompresser et observer le fichier `Makefile`.