Compilation séparée en OCaml

Rappels : compilation simple Un code OCaml peut être interprété (avec `utop` par exemple) ou bien compilé avec `ocamlc` (ou `ocamlopt`), en ajoutant des options si le code utilise la librairie `Graphics`.

```
ocamlc mon_code.ml -o mon_code.exe
ocamlfind ocamlopt -linkpkg -package graphics mon_code.ml -o mon_code.exe
```

Lancée avec l'option `-i`, la commande `ocamlc` affiche la liste des valeurs définies dans le fichier.

Exemple Fichier `mon_code.ml` :

```
1 let b = let a = 15 in a + 2
2 let deux() = 2
3 let _ = 28
```

Résultat de `ocamlc -i mon_code.ml` :

```
val b : int
val deux : unit -> int
```

Compilation séparée Toutes les valeurs (variables, fonctions, types...) définies dans un fichier nommé `a.ml` peuvent être utilisées dans un autre fichier `main.ml` comme si elles étaient définies dans un module nommé `A`. Il faut d'abord compiler le fichier `a.ml` sans édition de liens, cela produit le fichier `a.cmo` qui sera ensuite utilisé pour compiler le fichier `main.ml`.

Exemple Fichier `a.ml` :

```
1 let deux = 2
```

Fichier `b.ml` :

```
1 let deux = "deux"
```

Fichier `main.ml`

```
1 let main  =
2   Printf.printf "deux vaut %d\n" (A.deux);
3   Printf.printf "2 s'écrit %s\n" (B.deux)
```

```
ocamlc -c a.ml
ocamlc -c b.ml
ocamlc a.cmo b.cmo main.ml -o main.exe
```

Avec utop

- On peut utiliser `a.ml` ou `b.ml` seuls avec le classique `#use`.
- On peut utiliser `main.ml` après avoir chargé avec `#load` les fichiers `.cmo` préalablement compilés (avec `ocamlc -c a.ml` et `ocamlc -c b.ml` respectivement).

Différentes erreurs

- Si le fichier `a.cmo` n'est pas compilé, on obtient `Error: Unbound module A`. On obtient la même erreur sur utop si on fait `#use main.ml` sans avoir fait au préalable `#load a.ml`.
- Si le fichier `a.cmo` est compilé, présent dans le dossier, mais n'est pas indiqué dans la commande de compilation, cela ne génère aucune erreur, le fichier est bien pris en compte.
- Si les fichiers `a.mli`[a] et `b.mli` sont présents et indiqués dans la commande de compilation, l'absence des fichiers `.cmo` génère ce genre d'erreur.

```
Error: No implementation provided for the following modules:
         A referenced from Main (main.cmo)
         B referenced from Main (main.cmo)
```

- Si on oublie de passer les fichiers `.mli`, c'est comme s'ils n'étaient pas là (premier cas).
- Sur utop, si on oublie de faire `#load a.cmo`, la commande `#use main.ml` génère l'erreur `Error: Unbound module A`, tandis que si on a oublié de compilé le fichier `a.cmo`, la commande `#load a.cmo` génère l'erreur `Cannot find file a.cmo`.

a. *Cf.* la fiche-outil sur les interfaces en OCaml