Lancer deux fils d'exécution en parallèle en OCaml

Pour gérer des fils d'exécution en OCaml on utilise le module `Thread.Posix`. On peut charger ce module sur `utop` en tapant `#require "threads.posix";;`. Afin de compiler un programme OCaml utilisant ce module on pourra utiliser la ligne de commande ci-dessous.

```
ocamlc -thread unix.cma threads.cma main.ml -o main
```

Les fils d'exécution sont des objets de type `Thread.t`. Ils doivent être créés par appel à la fonction `Thread.create : ('a -> 'b) -> 'a -> Thread.t` qui prend en arguments :

- une fonction qui est celle que le fil d'exécution doit exécuter ;
- l'argument sur lequel ce fil d'exécution doit exécuter la fonction.

La fonction `Thread.create` retourne alors le fil d'exécution ainsi créé.

De même que pour la création de fil d'exécution en C, il est donc nécessaire de se munir d'une structure permettant la gestion des entrées/sorties de telles fonctions.
On peut demander à attendre qu'un fil d'exécution ait terminé son exécution grâce à la fonction `Thread.join : Thread.t -> unit` qui prend en argument le fil d'exécution en question.

```
type args =
  {
    nb: int           ;
    mutable res : int ;
  }

let au_carre (args: args): unit =
  args.res <- args.nb * args.nb
```

```
let () =
  let arg1 = {nb = 2; res = -1} in
  let arg2 = {nb = 9; res = -1} in
  let pa = Thread.create au_carre arg1 in
  let pb = Thread.create au_carre arg2 in
  Thread.join pa;
  Thread.join pb;
  assert (arg1.res = 4 && arg2.res = 81)
```