Entrées-sorties : Exemple

```
let copy_file f1 f2 =
  let c1 = open_in f1 in
  let c2 = open_out f2 in
  try
    while true do
      let v = input_char c1 in
      output_char c2 v
    done
  with End_of_file ->
    close_in c1; close_out c2

let () = copy_file Sys.argv.(1) Sys.argv.(2)
```

Entrées-sorties : canaux

- Les canaux d'entrée sont de type `in_channel`

```
# stdin;;
- : in_channel = <abstr>
```

- Les canaux de sortie sont de type `out_channel`

```
# stdout;;
- : out_channel = <abstr>
# stderr;;
- : out_channel = <abstr>
```

Entrées-sorties : lecture

- depuis l'entrée standard

```
read_line: unit -> string
read_int: unit -> int
```

- depuis n'importe quel canal d'entrée

```
input_char: in_channel -> char
input_line: in_channel -> string
input_byte: in_channel -> int
```

```
# let i = read_int ();;
10
val i : int = 10
# let c = input_char stdin;;
a
val c : char = 'a'
```

Entrées-sorties : écriture

- sur la sortie standard

```
print_char: string -> unit
print_int: int -> unit
print_string: string -> unit
```

- sur n'importe quel canal de sortie

```
output_char: out_channel -> char -> unit
output_string: out_channel -> string -> unit
```

```
# print_char 'a';;
a- : unit = ()
# output_string stdout "bonjour\n";;
bonjour
- : unit = ()
```

Entrées-sorties : Fichiers comme canaux

Les fichiers sont manipulés comme des canaux.
Ils doivent être ouverts en lecture ou en écriture :

```
open_in : string -> in_channel
```

```
open_out : string -> out_channel
```

```
# let cout = open_out "file.txt";;
val cout : out_channel = <abstr>
# output_string cout "bonjour\n";;
- : unit = ()
```

Ne pas oublier de fermer les canaux pour être sûr que tout soit bien écrit :

```
# close_out cout;;
- : unit = ()
```

Entrées-sorties : Fichiers comme canaux

```
# let cin = open_in "file.txt";;
val cin : in_channel = <abstr>
# input_char cin;;
- : char = 'b'
# close_in cin;;
- : unit = ()
```

Entrées-sorties : Sérialisation

On peut lire ou écrire des valeurs arbitraires dans des canaux

```
# let c = open_out "foo" in
  output_value c (1, 3.14, true);
  close_out c;;
- : unit = ()
```

Le format d'écriture est spécifique au langage OCaml

Il est préférable d'indiquer le type de la valeur lue

```
# let c = open_in "foo";;
val c : in_channel = <abstr>
# let v : int * float * bool = input_value c;;
val v : int * float * bool = (1, 3.14, true)
```

Entrées-sorties : Sérialisation

La lecture est non sûre :

```
# let c = open_in "foo";;
val c : in_channel = <abstr>
# let v = input_value c in fst (fst v)
```

```
Process caml-toplevel segmentation fault
```