Majeure 2
Rattrapage Caml
bis

Jean-Jacques Lévy

`jean-jacques.levy@inria.fr`

21 janvier 1998

Les modules de bibliothéques – int.mli

```
(* Operations on integers *)

(* Integers are 31 bits wide (or 63 bits on 64-bit processors).
   All operations are taken modulo $2^{31}$ (or $2^{63}$).
   They do not fail on overflow. *)

exception Division_by_zero;;

value minus : int -> int = 1 "~int"
  and minus_int : int -> int = 1 "~int"
        (* Unary negation. You can write [-e] instead of [minus e]. *)
  and succ : int -> int = 1 "succ"
        (* [succ x] is [x+1]. *)
  and pred : int -> int = 1 "pred"
        (* [pred x] is [x-1]. *)
  and prefix + : int -> int -> int = 2 "+int"
...;;
```

Les modules de bibliothéques – int.mli (2)

```
value string_of_int : int -> string
        (* Convert the given integer to its decimal
           representation. *)
  and int_of_string : string -> int = 1 "int_of_string"
        (* Convert the given string to an integer,
           in decimal (by default) or in hexadecimal,
           octal or binary if the string begins with
           [0x], [0o] or [0b].
           Raise [Failure "int_of_string"] if the given
           string is not a valid representation of an
           integer. *)
```

Les modules de bibliothéques – string.mli

`s.[n]` est le $n + 1$ème caractère de `s`

`s.[n] <- c` positionne ce caractère

Ce qui peut être trouvé dans l'interface `string.mli` .

```
value string_length : string -> int;;
value nth_char : string -> int -> char
  (* [nth_char s n] returns character number [n] in string [s].
     The first character is character number 0.
     The last character is character number [string_length s - 1].
     Raise [Invalid_argument "nth_char"] if [n] is ouside the range
     0 to [(string_length s - 1)].
     You can also write [s.[n]] instead of [nth_char s n]. *)
  and set_nth_char : string -> int -> char -> unit
  (* [set_nth_char s n c] modifies string [s] in place,
     replacing the character number [n] by [c].
     Raise [Invalid_argument "set_nth_char"] if [n] is ouside
     the range 0 to [(string_length s - 1)].
     You can also write [s.[n] <- c] instead of [set_nth_char s n c]. *)
;;
```

Les modules de bibliothéques – string.mli (2)

Et aussi plus succinctement:

```
value prefix ^ : string -> string -> string
  and sub_string : string -> int -> int -> string;;
value create_string : int -> string
  and make_string : int -> char -> string;;
value fill_string : string -> int -> int -> char -> unit
  and blit_string : string -> int -> string -> int -> int -> unit
  and replace_string : string -> string -> int -> unit;;
value eq_string : string -> string -> bool = 2 "=string"
  and neq_string : string -> string -> bool = 2 "<>string"
  and le_string : string -> string -> bool = 2 "<=string"
  and lt_string : string -> string -> bool = 2 "<string"
  and ge_string : string -> string -> bool = 2 ">=string"
  and gt_string : string -> string -> bool = 2 ">string";;
value compare_strings : string -> string -> int = 2 "compare_strings";;
value string_for_read : string -> string;;
```

Entrées/Sorties tamponnées – io.mli

```
type in_channel  and  out_channel;;
exception End_of_file;;
value stdin: in_channel  and  stdout: out_channel;;

value exit : int -> 'a;;

value print_char : char -> unit
  and print_string : string -> unit
  and print_int : int -> unit
  and print_float : float -> unit
  and print_endline : string -> unit
  and print_newline : unit -> unit;;

value read_line : unit -> string
  and read_int : unit -> int
  and read_float : unit -> float;;
```

Entrées/Sorties tamponnées – io.mli (2)

```
value open_out : string -> out_channel
  and flush : out_channel -> unit
  and output_char : out_channel -> char -> unit
  and output_string : out_channel -> string -> unit
  and output : out_channel -> string -> int -> int -> unit
  and seek_out : out_channel -> int -> unit
  and pos_out : out_channel -> int
  and close_out : out_channel -> unit;;

value open_in : string -> in_channel
  and input_char : in_channel -> char
  and input_line : in_channel -> string
  and input : in_channel -> string -> int -> int -> int
  and seek_in : in_channel -> int -> unit
  and pos_in : in_channel -> int
  and close_in : in_channel -> unit;;
```

Où trouver les modules de bibliothèques

En Unix, on peut lire sous Emacs tous les `.mli` en

`/usr/local/lib/caml-light`

Sur Mac ou Pc, dans le répertoire `lib` de la distribution Caml-light.

Sur le Web, on les trouve en:

`http://pauillac.inria.fr/caml/man-caml/`

aussi à l'X en:

Compilation séparée

2 types de noms d'identificateur:

nommage absolu: `graphics__moveto`, `graphics__lineto`, etc

nommage relatif: `moveto`, `lineto` après avoir fait `#open "graphics"`

Le top-level interactif est dans un module `top` implicite.

Pour charger un module, on fait `load "x.ml"`, ou `load_object "x.zo"`.

Les interfaces de module doivent être compilés (format `.zi`). L'interface d'un module est implicitement importé par le nommage absolu (contenant le nom du module) ou par la directive `#open`.

Compilation séparée (2)

Les suffixes `.zo` désignent des fichiers compilés d'implémentation.

La compilation génère des fichiers `.zo` ou `.zi` à partir de `.ml` ou `.mli`. Si pas de `.mli`, un `.zi` est automatiquement généré en faisant un interface standard. La compilation de `.mli`, puis de `.ml` permet de vérifier l'adéquation de l'interface par rapport à l'implémentation.