A typed foreign function interface for ML

Jeremy Yallop

26 November 2013

Itinerary

Background / using ctypes / inside ctypes

Itinerary

Background / using ctypes / inside ctypes

Foreign function interfaces

Function calls between (e.g.) ML and C

Different views of data

Integration between runtimes

Foreign function interfaces

Function calls between (e.g.) ML and C

Calling system libraries or other C code from an ML program. Registering ML functions as callbacks.

Different views of data

Integration between runtimes

Foreign function interfaces

Function calls between (e.g.) ML and C

Calling system libraries or other C code from an ML program. Registering ML functions as callbacks.

Different views of data

ML data is a tagged graph. C data is untagged and essentially flat.

Integration between runtimes

Foreign function interfaces

Function calls between (e.g.) ML and C

Calling system libraries or other C code from an ML program. Registering ML functions as callbacks.

Different views of data

ML data is a tagged graph. C data is untagged and essentially flat.

Integration between runtimes

GC vs manual memory management. Possibly different calling conventions, etc.

OCaml's FFI

A single value representation

Macros for accessing OCaml values

Macros for interacting with the GC

OCaml's FFI

A single value representation

Values are immediates or pointers to blocks, distinguished by the low bits.

Macros for accessing OCaml values

Macros for interacting with the GC

OCaml's FFI

A single value representation

Values are immediates or pointers to blocks, distinguished by the low bits.

Macros for accessing OCaml values

Macros (`Val_int` / `Int_val`) for converting between C integers and tagged integers. Macros and functions (`caml_alloc_string` / `String_val` &c.) for allocating/accessing blocks.

Macros for interacting with the GC

OCaml's FFI

A single value representation

Values are immediates or pointers to blocks, distinguished by the low bits.

Macros for accessing OCaml values

Macros (`Val_int` / `Int_val`) for converting between C integers and tagged integers. Macros and functions (`caml_alloc_string` / `String_val` &c.) for allocating/accessing blocks.

Macros for interacting with the GC

Macros (`CAMLParam*` / `CAMLreturn`) for registering/unregistering local values with the runtime.

OCaml's FFI: pitfalls

An example C stub

OCaml's FFI: pitfalls

```
char *first_line(size_t max_bytes, const char *filename)
{
   char *buf = malloc(max_bytes);
   if (buf == NULL) return NULL;

   FILE *fp = fopen(filename, "r");
   if (fp != NULL) {
      fgets(buf, max_bytes, fp);
      fclose(fp);
   }
   else { free(buf); buf = NULL; }

   return buf;
}
```

C code

OCaml's FFI: pitfalls

```
value first_line(value max_bytes, value filename)
{
   char *buf = malloc(max_bytes);
   if (buf == NULL) return NULL;

   FILE *fp = fopen(filename, "r");
   if (fp != NULL) {
      fgets(buf, max_bytes, fp);
      fclose(fp);
   }
   else { free(buf); buf = NULL; }

   return buf;
}
```

Parameters/return values become `value`

OCaml's FFI: pitfalls

```
value first_line(value max_bytes, value filename)
{
  CAMLparam2(filename, max_bytes);
  char *buf = malloc(max_bytes);
  if (buf == NULL) return NULL;

  FILE *fp = fopen(filename, "r");
  if (fp != NULL) {
      fgets(buf, max_bytes, fp);
      fclose(fp);
  }
  else { free(buf); buf = NULL; }

  CAMLreturn(buf);
}
```

Add GC hooks for parameters

OCaml's FFI: pitfalls

```
value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(filename, "r");
   if (fp != NULL) {
      fgets(buf, max_bytes, fp);
      fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Allocate the buffer in the OCaml heap

OCaml's FFI: pitfalls

```
value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   const char *c_filename = String_val(filename);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(c_filename, "r");
   if (fp != NULL) {
      fgets(String_val(buf), max_bytes, fp);
      fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Extract string addresses to pass to C

OCaml's FFI: pitfalls

```
external first_line : string -> int -> string = "first_line"
```

```
value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   const char *c_filename = String_val(filename);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(c_filename, "r");
   if (fp != NULL) {
      fgets(String_val(buf), max_bytes, fp);
      fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Add an OCaml declaration

OCaml's FFI: pitfalls

```
external first_line : string -> int -> string = "first_line"

value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   const char *c_filename = String_val(filename);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(c_filename, "r");
   if (fp != NULL) {
       fgets(String_val(buf), max_bytes, fp);
       fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Compiles successfully!

OCaml's FFI: pitfalls

```
external first_line : string -> int -> string = "first_line"

value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   const char *c_filename = String_val(filename);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(c_filename, "r");
   if (fp != NULL) {
       fgets(String_val(buf), max_bytes, fp);
       fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Bug: parameters interchanged (crash!)

OCaml's FFI: pitfalls

```
external first_line : string -> int -> string = "first_line"

value first_line(value max_bytes, value filename)
{
   CAMLparam2(filename, max_bytes);
   const char *c_filename = String_val(filename);
   CAMLlocal1(buf);
   buf = caml_alloc_string(max_bytes);

   FILE *fp = fopen(c_filename, "r");
   if (fp != NULL) {
      fgets(String_val(buf), max_bytes, fp);
      fclose(fp);
   }
   else failwith("fopen failed");

   CAMLreturn(buf);
}
```

Bug: invalidated pointer (crash?)

OCaml's FFI: pitfalls

```
external first_line : string -> int -> string = "first_line"

value first_line(value max_bytes, value filename)
{
  CAMLparam2(filename, max_bytes);
  const char *c_filename = String_val(filename);
  CAMLlocal1(buf);
  buf = caml_alloc_string(max_bytes);

  FILE *fp = fopen(c_filename, "r");
  if (fp != NULL) {
     fgets(String_val(buf), max_bytes, fp);
     fclose(fp);
  }
  else failwith("fopen failed");

  CAMLreturn(buf);
}
```

Bug: missing conversion (misbehaviour)

Itinerary

~~Background~~ / using ctypes / inside ctypes

Outline of the ctypes approach

OCaml, not C

Types, not values

“What,” not “how”

Outline of the ctypes approach

OCaml, not C

Access C values from OCaml, not vice-versa. Why? abstraction, type safety, automatic memory management, ...

Types, not values

“What,” not “how”

Outline of the ctypes approach

OCaml, not C

Access C values from OCaml, not vice-versa. Why? abstraction, type safety, automatic memory management, ...

Types, not values

Types are sufficient to determine the interface.

“What,” not “how”

Outline of the ctypes approach

OCaml, not C

Access C values from OCaml, not vice-versa. Why? abstraction, type safety, automatic memory management, ...

Types, not values

Types are sufficient to determine the interface.

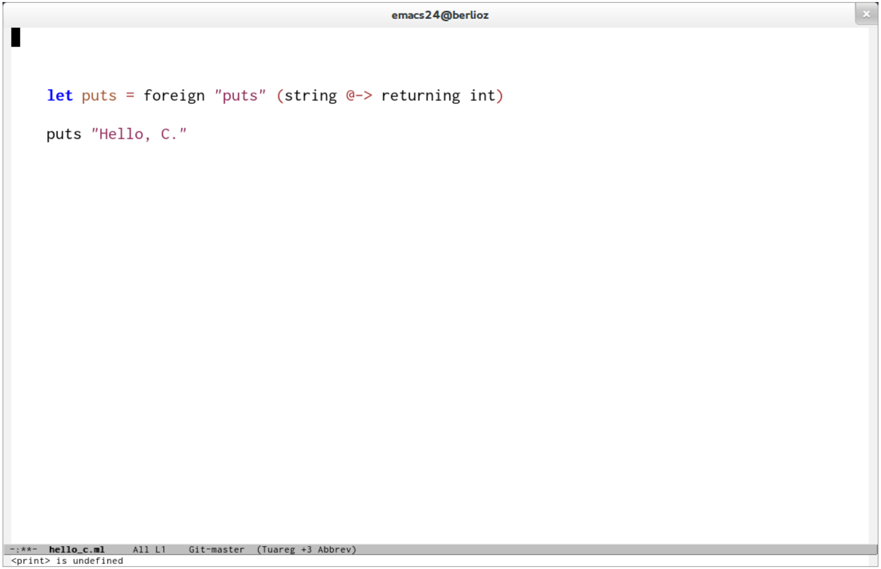
“What,” not “how”

Build a typed embedded DSL, separating construction from interpretation.

Ctypes in action

[demo: hello, world]

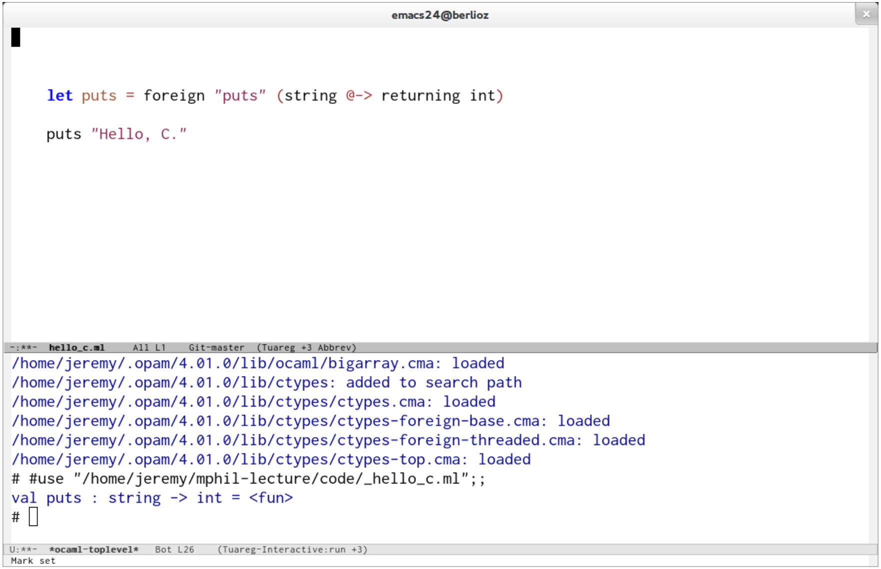
Ctypes in action



```
let puts = foreign "puts" (string @-> returning int)

puts "Hello, C."
```

```
-:**-  hello_c.ml    All L1    Git-master  (Tuareg +3 Abbrev)
<print> is undefined
```

Ctypes in action



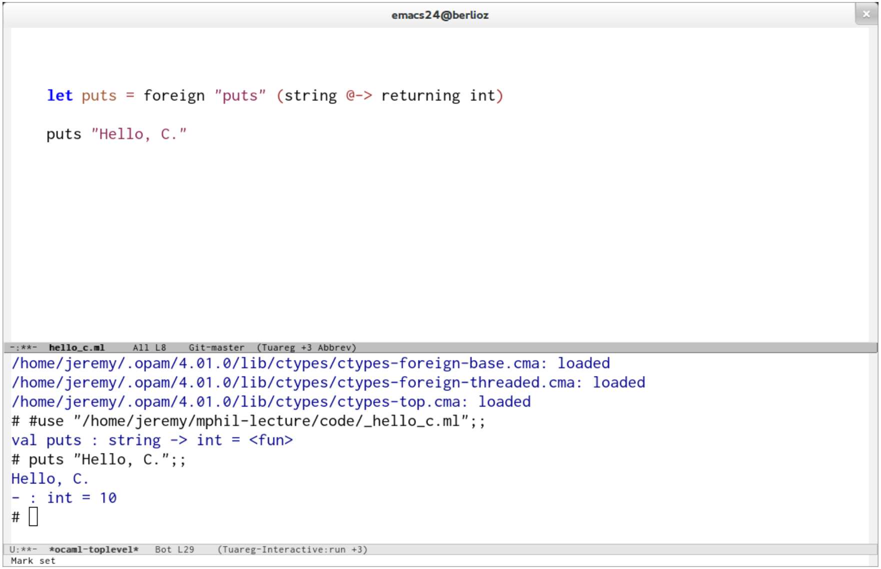
```
let puts = foreign "puts" (string @-> returning int)

puts "Hello, C."
```

-:**- hello_c.ml   All L1   Git-master   (Tuareg +3 Abbrev)

```
/home/jeremy/.opam/4.01.0/lib/ocaml/bigarray.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes: added to search path
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-foreign-base.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-foreign-threaded.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-top.cma: loaded
# #use "/home/jeremy/mphil-lecture/code/_hello_c.ml";;
val puts : string -> int = <fun>
# 
```

U:**- *ocaml-toplevel*   Bot L26   (Tuareg-Interactive:run +3)
Mark set

Ctypes in action



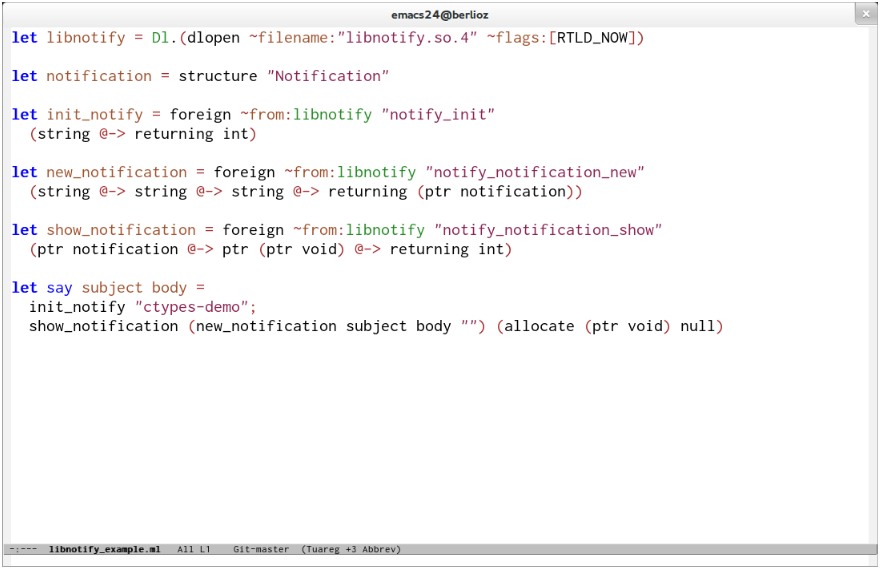
```
let puts = foreign "puts" (string @-> returning int)

puts "Hello, C."
```

```
-:**-  hello_c.ml      All L8     Git-master   (Tuareg +3 Abbrev)
```

```
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-foreign-base.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-foreign-threaded.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes/ctypes-top.cma: loaded
# #use "/home/jeremy/mphil-lecture/code/_hello_c.ml";;
val puts : string -> int = <fun>
# puts "Hello, C.";;
Hello, C.
- : int = 10
# 
```

```
U:**-  *ocaml-toplevel*   Bot L29    (Tuareg-Interactive:run +3)
Mark set
```

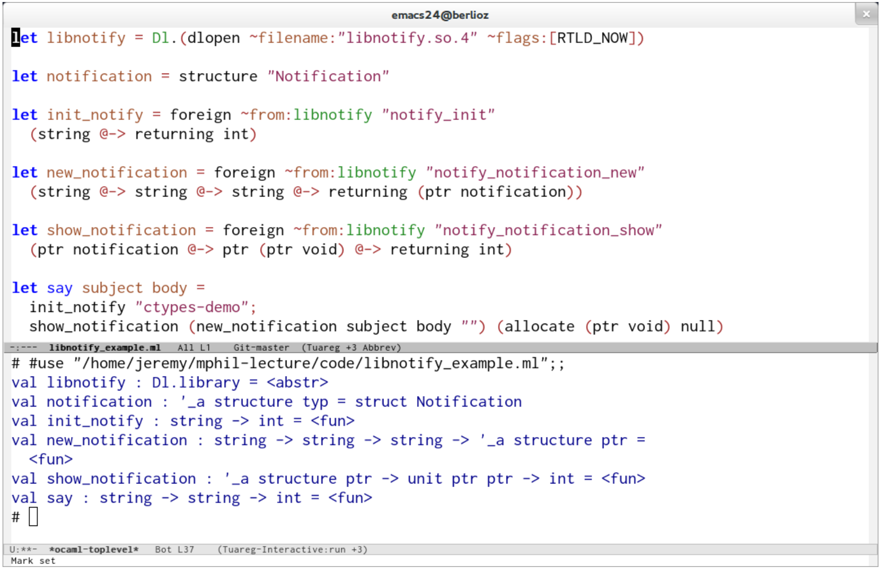
Ctypes in action



```
let libnotify = Dl.(dlopen ~filename:"libnotify.so.4" ~flags:[RTLD_NOW])

let notification = structure "Notification"

let init_notify = foreign ~from:libnotify "notify_init"
  (string @-> returning int)

let new_notification = foreign ~from:libnotify "notify_notification_new"
  (string @-> string @-> string @-> returning (ptr notification))

let show_notification = foreign ~from:libnotify "notify_notification_show"
  (ptr notification @-> ptr (ptr void) @-> returning int)

let say subject body =
  init_notify "ctypes-demo";
  show_notification (new_notification subject body "") (allocate (ptr void) null)
```

-:--- libnotify_example.ml   All L1   Git-master (Tuareg +3 Abbrev)

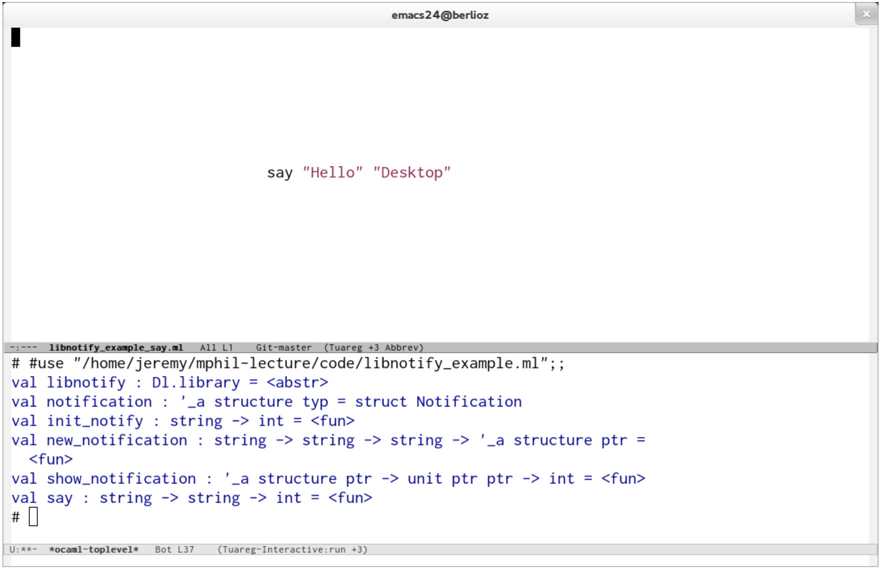
Ctypes in action



emacs24@berlioz

```
let libnotify = Dl.(dlopen ~filename:"libnotify.so.4" ~flags:[RTLD_NOW])

let notification = structure "Notification"

let init_notify = foreign ~from:libnotify "notify_init"
  (string @-> returning int)

let new_notification = foreign ~from:libnotify "notify_notification_new"
  (string @-> string @-> string @-> returning (ptr notification))

let show_notification = foreign ~from:libnotify "notify_notification_show"
  (ptr notification @-> ptr (ptr void) @-> returning int)

let say subject body =
  init_notify "ctypes-demo";
  show_notification (new_notification subject body "") (allocate (ptr void) null)
-:---  libnotify_example.ml   All L1    Git-master  (Tuareg +3 Abbrev)
# #use "/home/jeremy/mphil-lecture/code/libnotify_example.ml";;
val libnotify : Dl.library = <abstr>
val notification : '_a structure typ = struct Notification
val init_notify : string -> int = <fun>
val new_notification : string -> string -> string -> '_a structure ptr =
  <fun>
val show_notification : '_a structure ptr -> unit ptr ptr -> int = <fun>
val say : string -> string -> int = <fun>
# 
U:**-  *ocaml-toplevel*   Bot L37    (Tuareg-Interactive:run +3)
Mark set
```

Ctypes in action



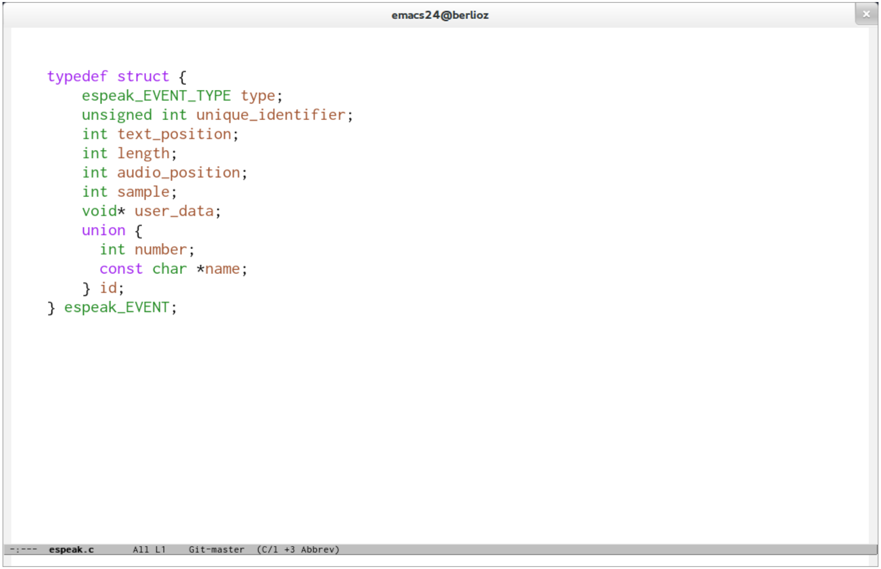
```
say "Hello" "Desktop"
```

-:--- libnotify_example_say.ml   All L1   Git-master (Tuareg +3 Abbrev)

```
# #use "/home/jeremy/mphil-lecture/code/libnotify_example.ml";;
val libnotify : Dl.library = <abstr>
val notification : '_a structure typ = struct Notification
val init_notify : string -> int = <fun>
val new_notification : string -> string -> string -> '_a structure ptr =
  <fun>
val show_notification : '_a structure ptr -> unit ptr ptr -> int = <fun>
val say : string -> string -> int = <fun>
# 
```

U:**- *ocaml-toplevel*   Bot L37   (Tuareg-Interactive:run +3)

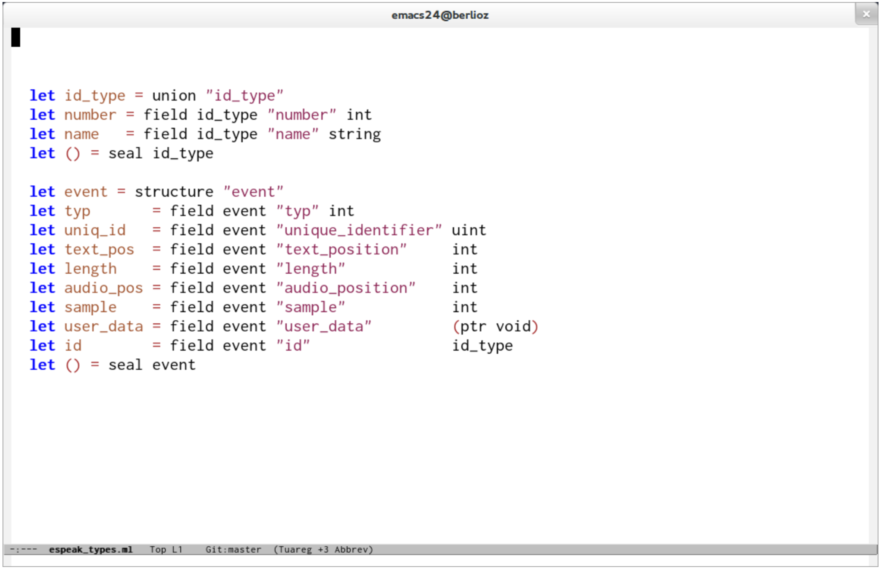
Ctypes in action



```
typedef struct {
    espeak_EVENT_TYPE type;
    unsigned int unique_identifier;
    int text_position;
    int length;
    int audio_position;
    int sample;
    void* user_data;
    union {
      int number;
      const char *name;
    } id;
} espeak_EVENT;
```

```
-:---  espeak.c      All L1    Git-master  (C/l +3 Abbrev)
```

Ctypes in action

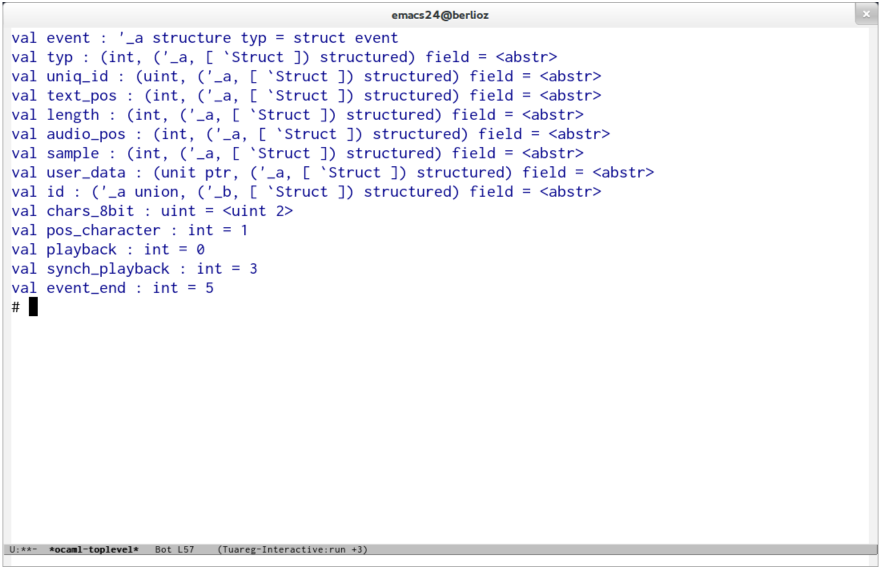


emacs24@berlioz

```
let id_type = union "id_type"
let number = field id_type "number" int
let name   = field id_type "name" string
let () = seal id_type

let event = structure "event"
let typ       = field event "typ" int
let uniq_id   = field event "unique_identifier" uint
let text_pos  = field event "text_position"     int
let length    = field event "length"            int
let audio_pos = field event "audio_position"    int
let sample    = field event "sample"            int
let user_data = field event "user_data"         (ptr void)
let id        = field event "id"                id_type
let () = seal event
```

```
-:---  espeak_types.ml   Top L1    Git:master  (Tuareg +3 Abbrev)
```

Ctypes in action

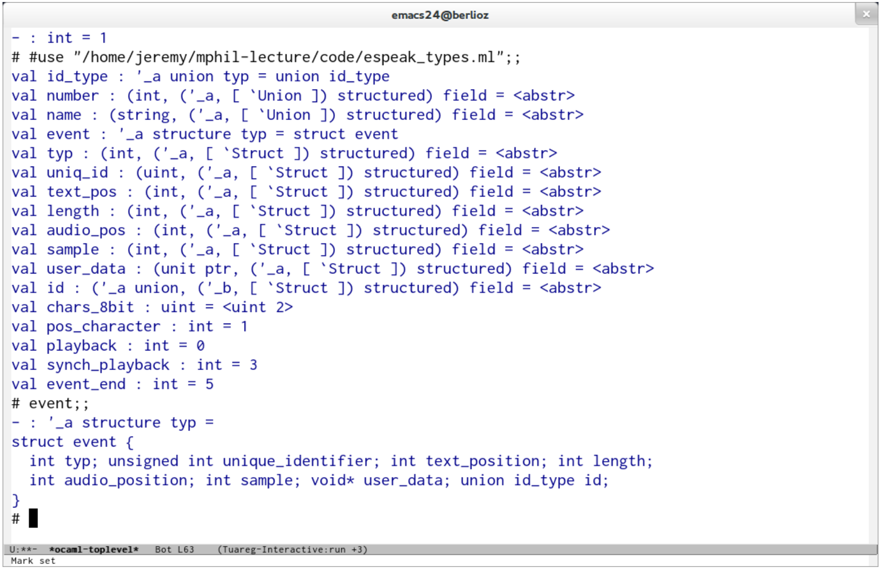


emacs24@berlioz

```
val event : '_a structure typ = struct event
val typ : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val uniq_id : (uint, ('_a, [ `Struct ]) structured) field = <abstr>
val text_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val length : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val audio_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val sample : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val user_data : (unit ptr, ('_a, [ `Struct ]) structured) field = <abstr>
val id : ('_a union, ('_b, [ `Struct ]) structured) field = <abstr>
val chars_8bit : uint = <uint 2>
val pos_character : int = 1
val playback : int = 0
val synch_playback : int = 3
val event_end : int = 5
# 
```

```
U:**-  *ocaml-toplevel*   Bot L57    (Tuareg-Interactive:run +3)
```

Ctypes in action

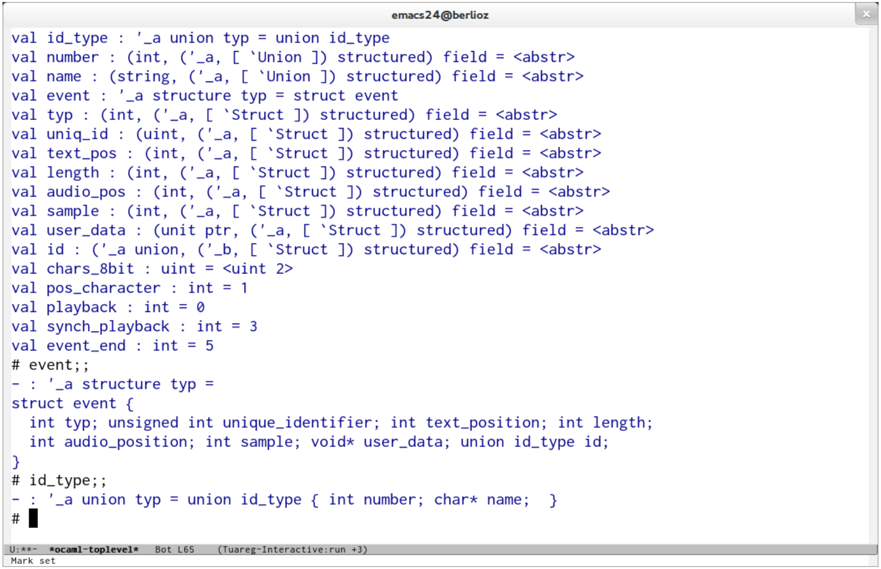


emacs24@berlioz

```
- : int = 1
# #use "/home/jeremy/mphil-lecture/code/espeak_types.ml";;
val id_type : '_a union typ = union id_type
val number : (int, ('_a, [ `Union ]) structured) field = <abstr>
val name : (string, ('_a, [ `Union ]) structured) field = <abstr>
val event : '_a structure typ = struct event
val typ : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val uniq_id : (uint, ('_a, [ `Struct ]) structured) field = <abstr>
val text_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val length : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val audio_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val sample : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val user_data : (unit ptr, ('_a, [ `Struct ]) structured) field = <abstr>
val id : ('_a union, ('_b, [ `Struct ]) structured) field = <abstr>
val chars_8bit : uint = <uint 2>
val pos_character : int = 1
val playback : int = 0
val synch_playback : int = 3
val event_end : int = 5
# event;;
- : '_a structure typ =
struct event {
  int typ; unsigned int unique_identifier; int text_position; int length;
  int audio_position; int sample; void* user_data; union id_type id;
}
# 
```

```
U:**-  *ocaml-toplevel*   Bot L63    (Tuareg-Interactive:run +3)
Mark set
```

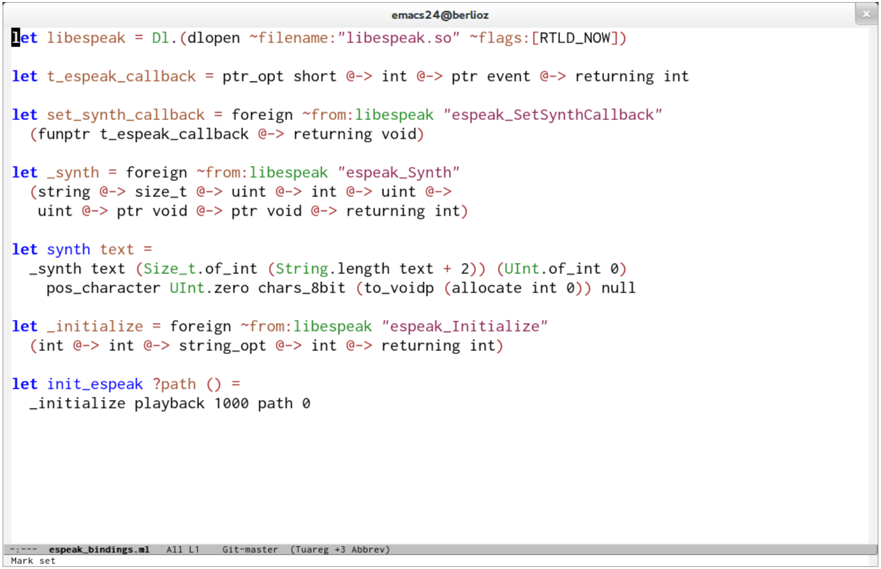
Ctypes in action


emacs24@berlioz

```
val id_type : '_a union typ = union id_type
val number : (int, ('_a, [ `Union ]) structured) field = <abstr>
val name : (string, ('_a, [ `Union ]) structured) field = <abstr>
val event : '_a structure typ = struct event
val typ : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val uniq_id : (uint, ('_a, [ `Struct ]) structured) field = <abstr>
val text_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val length : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val audio_pos : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val sample : (int, ('_a, [ `Struct ]) structured) field = <abstr>
val user_data : (unit ptr, ('_a, [ `Struct ]) structured) field = <abstr>
val id : ('_a union, ('_b, [ `Struct ]) structured) field = <abstr>
val chars_8bit : uint = <uint 2>
val pos_character : int = 1
val playback : int = 0
val synch_playback : int = 3
val event_end : int = 5
# event;;
- : '_a structure typ =
struct event {
  int typ; unsigned int unique_identifier; int text_position; int length;
  int audio_position; int sample; void* user_data; union id_type id;
}
# id_type;;
- : '_a union typ = union id_type { int number; char* name;  }
# 
```

```
U:**-  *ocaml-toplevel*   Bot L65    (Tuareg-Interactive:run +3)
Mark set
```

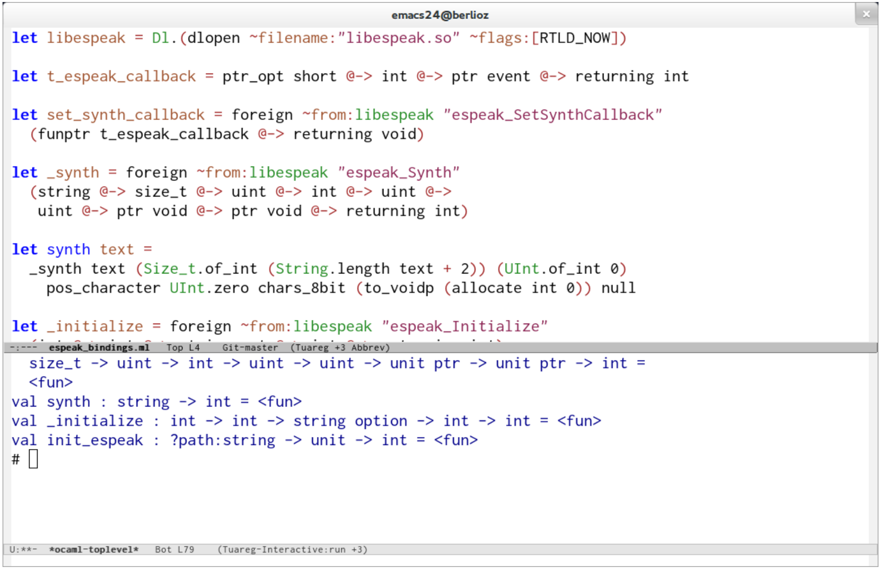
Ctypes in action



```
let libespeak = Dl.(dlopen ~filename:"libespeak.so" ~flags:[RTLD_NOW])

let t_espeak_callback = ptr_opt short @-> int @-> ptr event @-> returning int

let set_synth_callback = foreign ~from:libespeak "espeak_SetSynthCallback"
  (funptr t_espeak_callback @-> returning void)

let _synth = foreign ~from:libespeak "espeak_Synth"
  (string @-> size_t @-> uint @-> int @-> uint @->
   uint @-> ptr void @-> ptr void @-> returning int)

let synth text =
  _synth text (Size_t.of_int (String.length text + 2)) (UInt.of_int 0)
    pos_character UInt.zero chars_8bit (to_voidp (allocate int 0)) null

let _initialize = foreign ~from:libespeak "espeak_Initialize"
  (int @-> int @-> string_opt @-> int @-> returning int)

let init_espeak ?path () =
  _initialize playback 1000 path 0
```

```
-:---  espeak_bindings.ml   All L1    Git-master  (Tuareg +3 Abbrev)
Mark set
```

Ctypes in action



emacs24@berlioz

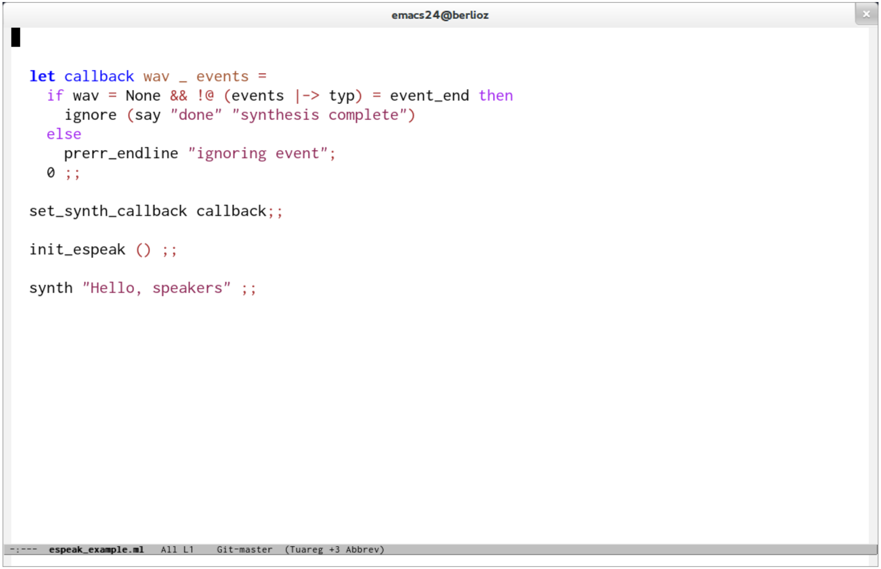
```
let libespeak = Dl.(dlopen ~filename:"libespeak.so" ~flags:[RTLD_NOW])

let t_espeak_callback = ptr_opt short @-> int @-> ptr event @-> returning int

let set_synth_callback = foreign ~from:libespeak "espeak_SetSynthCallback"
  (funptr t_espeak_callback @-> returning void)

let _synth = foreign ~from:libespeak "espeak_Synth"
  (string @-> size_t @-> uint @-> int @-> uint @->
   uint @-> ptr void @-> ptr void @-> returning int)

let synth text =
  _synth text (Size_t.of_int (String.length text + 2)) (UInt.of_int 0)
    pos_character UInt.zero chars_8bit (to_voidp (allocate int 0)) null

let _initialize = foreign ~from:libespeak "espeak_Initialize"
```

```
-:---  espeak_bindings.ml   Top L4    Git-master  (Tuareg +3 Abbrev)
```

```
  size_t -> uint -> int -> uint -> uint -> unit ptr -> unit ptr -> int =
  <fun>
val synth : string -> int = <fun>
val _initialize : int -> int -> string option -> int -> int = <fun>
val init_espeak : ?path:string -> unit -> int = <fun>
# 
```

```
U:**-  *ocaml-toplevel*   Bot L79    (Tuareg-Interactive:run +3)
```

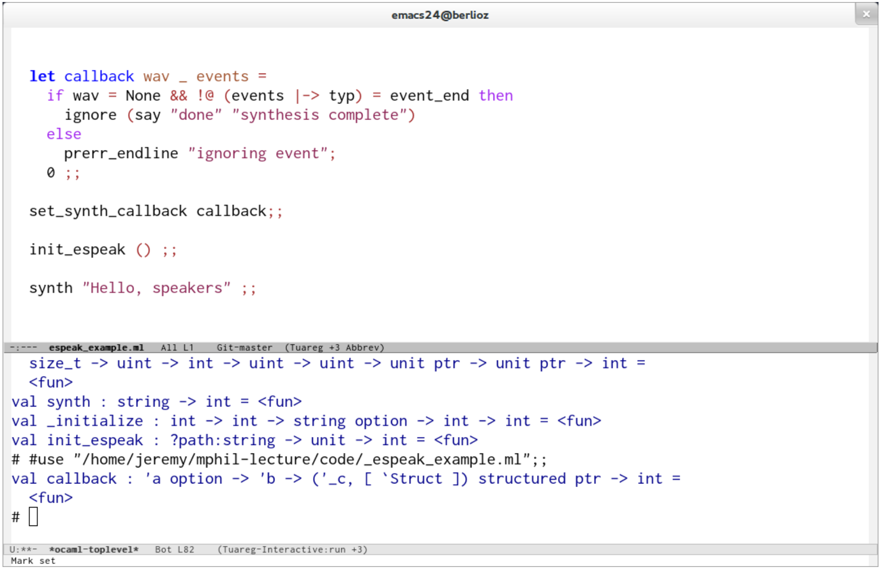
Ctypes in action


emacs24@berlioz

```
let callback wav _ events =
  if wav = None && !@ (events |-> typ) = event_end then
    ignore (say "done" "synthesis complete")
  else
    prerr_endline "ignoring event";
  0 ;;

set_synth_callback callback;;

init_espeak () ;;

synth "Hello, speakers" ;;
```

```
-:---  espeak_example.ml   All L1    Git-master  (Tuareg +3 Abbrev)
```

Ctypes in action



emacs24@berlioz

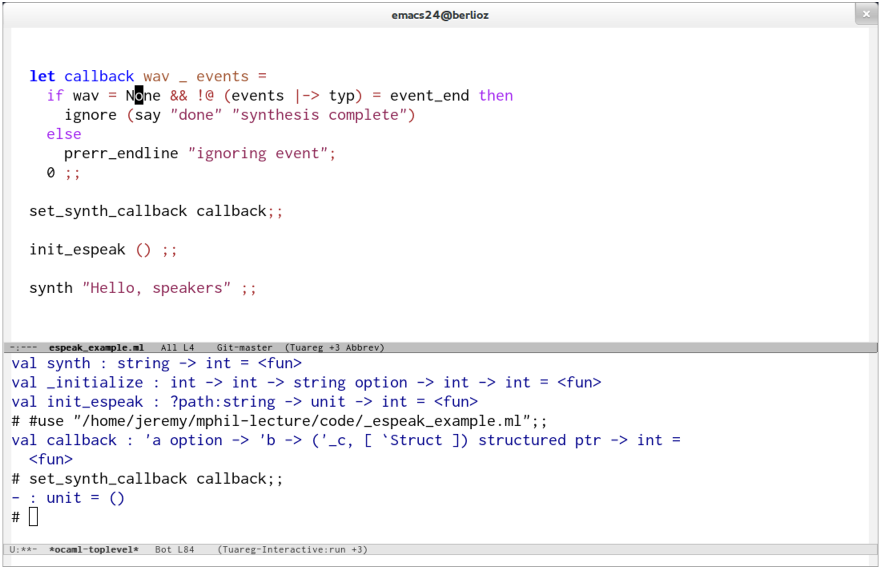
```
let callback wav _ events =
  if wav = None && !@ (events |-> typ) = event_end then
    ignore (say "done" "synthesis complete")
  else
    prerr_endline "ignoring event";
  0 ;;

set_synth_callback callback;;

init_espeak () ;;

synth "Hello, speakers" ;;
```

```
-:---  espeak_example.ml    All L1    Git-master  (Tuareg +3 Abbrev)
```

```
  size_t -> uint -> int -> uint -> uint -> unit ptr -> unit ptr -> int =
  <fun>
val synth : string -> int = <fun>
val _initialize : int -> int -> string option -> int -> int = <fun>
val init_espeak : ?path:string -> unit -> int = <fun>
# #use "/home/jeremy/mphil-lecture/code/_espeak_example.ml";;
val callback : 'a option -> 'b -> ('_c, [ `Struct ]) structured ptr -> int =
  <fun>
# 
```

```
U:**-  *ocaml-toplevel*   Bot L82    (Tuareg-Interactive:run +3)
Mark set
```

Ctypes in action



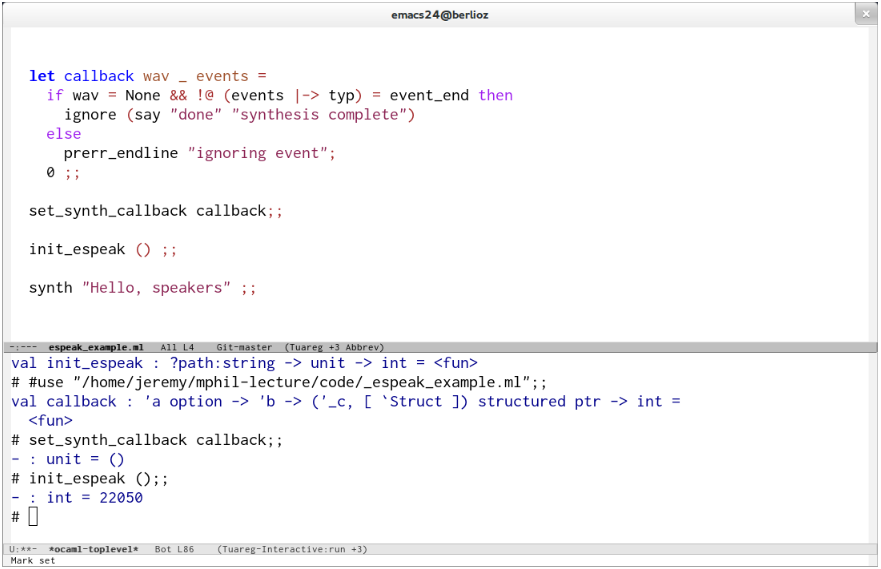
```
let callback wav _ events =
  if wav = None && !@ (events |-> typ) = event_end then
    ignore (say "done" "synthesis complete")
  else
    prerr_endline "ignoring event";
  0 ;;

set_synth_callback callback;;

init_espeak () ;;

synth "Hello, speakers" ;;
```

-:--- espeak_example.ml   All L4   Git-master (Tuareg +3 Abbrev)

```
val synth : string -> int = <fun>
val _initialize : int -> int -> string option -> int -> int = <fun>
val init_espeak : ?path:string -> unit -> int = <fun>
# #use "/home/jeremy/mphil-lecture/code/_espeak_example.ml";;
val callback : 'a option -> 'b -> ('_c, [ `Struct ]) structured ptr -> int =
  <fun>
# set_synth_callback callback;;
- : unit = ()
# 
```

U:**- *ocaml-toplevel*   Bot L84   (Tuareg-Interactive:run +3)

Ctypes in action



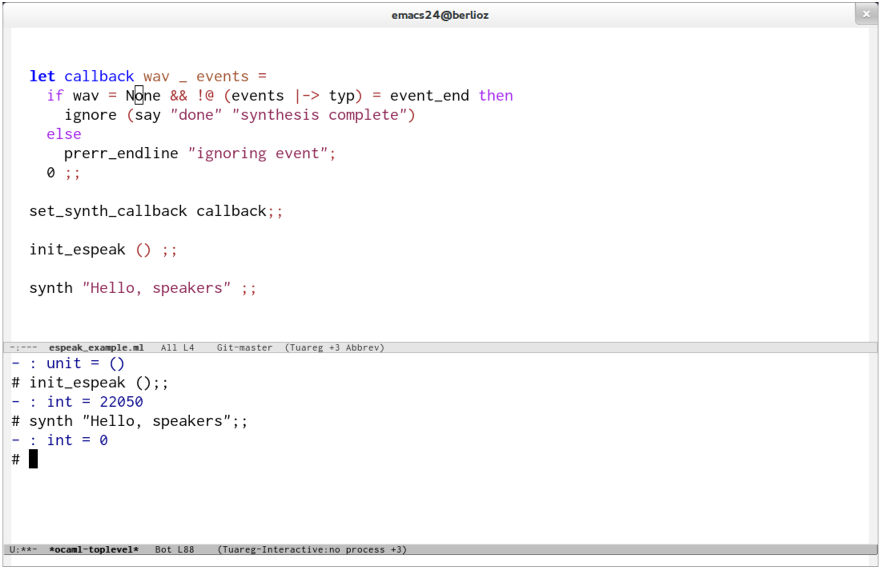
```
let callback wav _ events =
  if wav = None && !@ (events |-> typ) = event_end then
    ignore (say "done" "synthesis complete")
  else
    prerr_endline "ignoring event";
  0 ;;

set_synth_callback callback;;

init_espeak () ;;

synth "Hello, speakers" ;;
```

```
-:---  espeak_example.ml   All L4    Git-master  (Tuareg +3 Abbrev)
```

```
val init_espeak : ?path:string -> unit -> int = <fun>
# #use "/home/jeremy/mphil-lecture/code/_espeak_example.ml";;
val callback : 'a option -> 'b -> ('_c, [ `Struct ]) structured ptr -> int =
  <fun>
# set_synth_callback callback;;
- : unit = ()
# init_espeak ();;
- : int = 22050
# 
```

```
U:**-  *ocaml-toplevel*   Bot L86    (Tuareg-Interactive:run +3)
Mark set
```

Ctypes in action



emacs24@berlioz

```
let callback wav _ events =
  if wav = None && !@ (events |-> typ) = event_end then
    ignore (say "done" "synthesis complete")
  else
    prerr_endline "ignoring event";
  0 ;;

set_synth_callback callback;;

init_espeak () ;;

synth "Hello, speakers" ;;
```

```
-:---  espeak_example.ml   All L4    Git-master  (Tuareg +3 Abbrev)
```

```
- : unit = ()
# init_espeak ();;
- : int = 22050
# synth "Hello, speakers";;
- : int = 0
# 
```

```
U:**-  *ocaml-toplevel*   Bot L88    (Tuareg-Interactive:no process +3)
```

Embedded DSLs

Tailored to a specific domain

Host language functions for building terms

Host language types for typing terms

Separate building terms from interpretation

Embedded DSLs

Tailored to a specific domain

Parsing, database queries, music, financial contracts, graphics, &c.

Host language functions for building terms

Host language types for typing terms

Separate building terms from interpretation

Embedded DSLs

Tailored to a specific domain

Parsing, database queries, music, financial contracts, graphics, &c.

Host language functions for building terms

Can also borrow host language binding constructs.

Host language types for typing terms

Separate building terms from interpretation

Embedded DSLs

Tailored to a specific domain

Parsing, database queries, music, financial contracts, graphics, &c.

Host language functions for building terms

Can also borrow host language binding constructs.

Host language types for typing terms

For example, use a subset of ML types to type SQL tables or C types.

Separate building terms from interpretation

Embedded DSLs

Tailored to a specific domain

Parsing, database queries, music, financial contracts, graphics, &c.

Host language functions for building terms

Can also borrow host language binding constructs.

Host language types for typing terms

For example, use a subset of ML types to type SQL tables or C types.

Separate building terms from interpretation

The meaning of “declarative.” Allows multiple interpretations.

Embedded DSLs: multiple interpretations in ctypes

Interpretation

Compilation

Multi-process implementation

etc.

Embedded DSLs: multiple interpretations in ctypes

Interpretation

Dynamic binding, dynamic call construction. Interactive, but with interpretative overhead and some loss of safety.

Compilation

Multi-process implementation

etc.

Embedded DSLs: multiple interpretations in ctypes

Interpretation

Dynamic binding, dynamic call construction. Interactive, but with interpretative overhead and some loss of safety.

Compilation

Generation of C stubs from ctypes values. Type safe and efficient but with some complexity in the build system.

Multi-process implementation

etc.

Embedded DSLs: multiple interpretations in ctypes

Interpretation

Dynamic binding, dynamic call construction. Interactive, but with interpretative overhead and some loss of safety.

Compilation

Generation of C stubs from ctypes values. Type safe and efficient but with some complexity in the build system.

Multi-process implementation

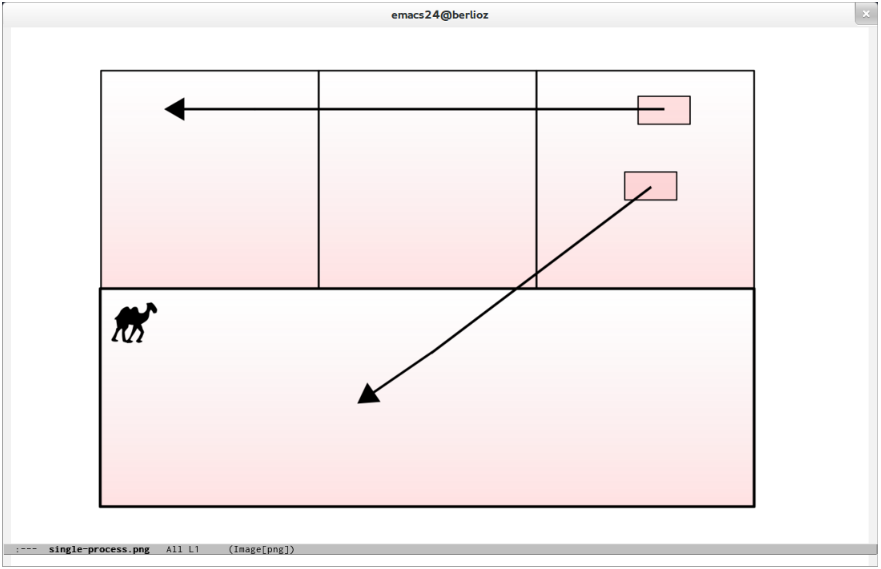
Sandbox C libraries to contain memory corruption. Intriguing possibilities: fork-based debugging, improved parallelism, . . .

etc.

CTYPES BACK IN ACTION

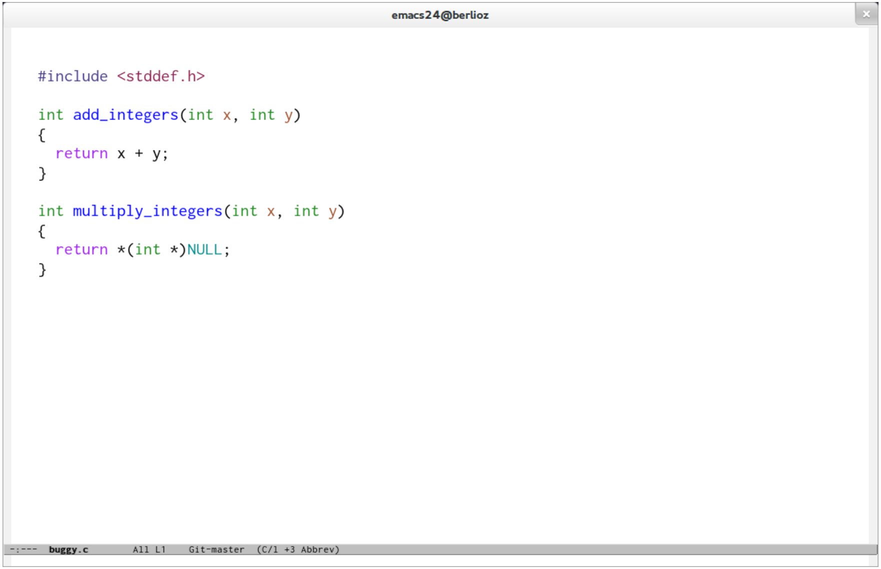
[demo: multi-process implementation]

CTYPES BACK IN ACTION



```
:---  single-process.png   All L1     (Image[png])
```

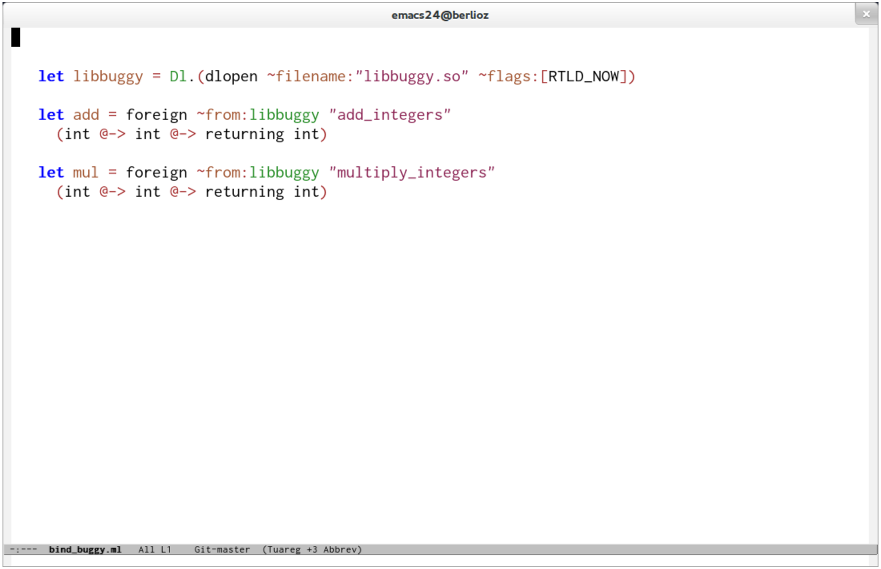
CTYPES BACK IN ACTION


emacs24@berlioz

```c
#include <stddef.h>

int add_integers(int x, int y)
{
  return x + y;
}

int multiply_integers(int x, int y)
{
  return *(int *)NULL;
}
```

```
-:---  buggy.c        All L1     Git-master  (C/l +3 Abbrev)
```

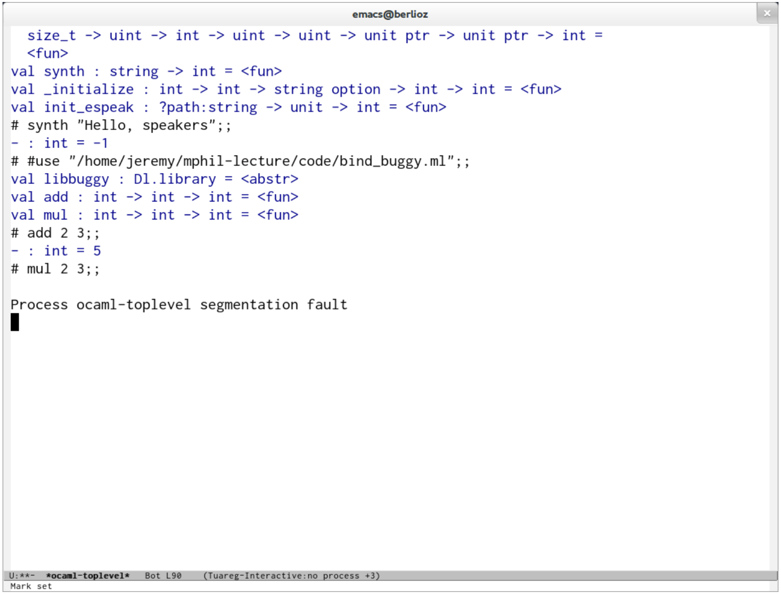
CTYPES BACK IN ACTION



```
let libbuggy = Dl.(dlopen ~filename:"libbuggy.so" ~flags:[RTLD_NOW])

let add = foreign ~from:libbuggy "add_integers"
  (int @-> int @-> returning int)

let mul = foreign ~from:libbuggy "multiply_integers"
  (int @-> int @-> returning int)
```

```
-:---  bind_buggy.ml   All L1    Git-master  (Tuareg +3 Abbrev)
```

CTYPES BACK IN ACTION

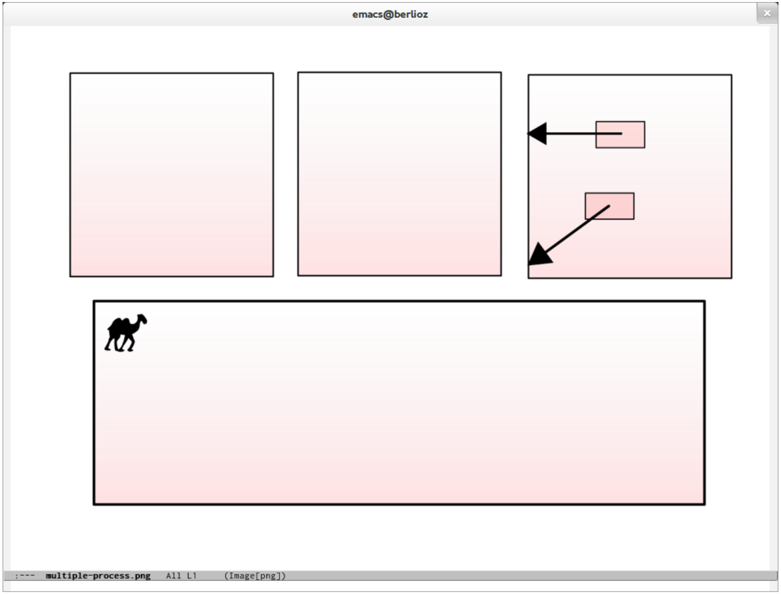


emacs@berlioz

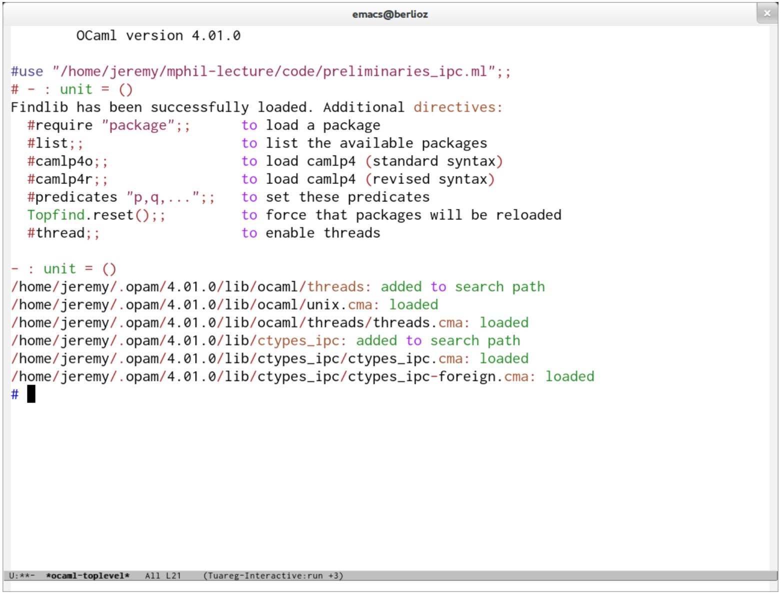
```
  size_t -> uint -> int -> uint -> uint -> unit ptr -> unit ptr -> int =
  <fun>
val synth : string -> int = <fun>
val _initialize : int -> int -> string option -> int -> int = <fun>
val init_espeak : ?path:string -> unit -> int = <fun>
# synth "Hello, speakers";;
- : int = -1
# #use "/home/jeremy/mphil-lecture/code/bind_buggy.ml";;
val libbuggy : Dl.library = <abstr>
val add : int -> int -> int = <fun>
val mul : int -> int -> int = <fun>
# add 2 3;;
- : int = 5
# mul 2 3;;

Process ocaml-toplevel segmentation fault
```

```
U:**-  *ocaml-toplevel*   Bot L90    (Tuareg-Interactive:no process +3)
Mark set
```

CTYPES BACK IN ACTION



CTYPES BACK IN ACTION

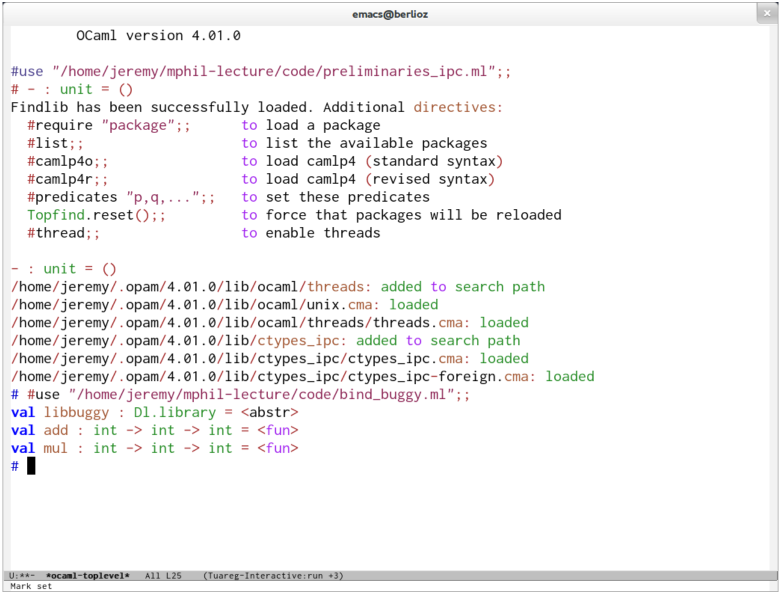


emacs@berlioz

```
        OCaml version 4.01.0

#use "/home/jeremy/mphil-lecture/code/preliminaries_ipc.ml";;
# - : unit = ()
Findlib has been successfully loaded. Additional directives:
  #require "package";;      to load a package
  #list;;                   to list the available packages
  #camlp4o;;                to load camlp4 (standard syntax)
  #camlp4r;;                to load camlp4 (revised syntax)
  #predicates "p,q,...";;   to set these predicates
  Topfind.reset();;         to force that packages will be reloaded
  #thread;;                 to enable threads

- : unit = ()
/home/jeremy/.opam/4.01.0/lib/ocaml/threads: added to search path
/home/jeremy/.opam/4.01.0/lib/ocaml/unix.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ocaml/threads/threads.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc: added to search path
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc-foreign.cma: loaded
# 
```

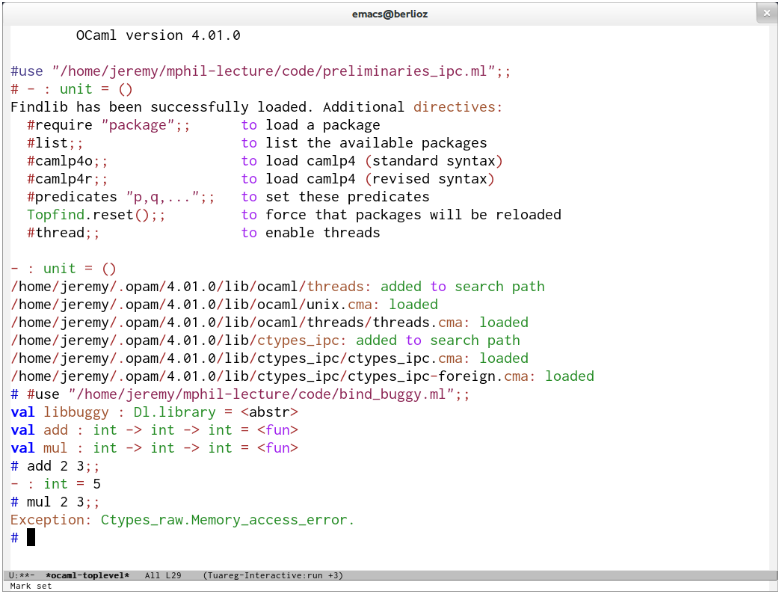
```
U:**-  *ocaml-toplevel*   All L21    (Tuareg-Interactive:run +3)
```

CTYPES BACK IN ACTION



```
emacs@berlioz
        OCaml version 4.01.0

#use "/home/jeremy/mphil-lecture/code/preliminaries_ipc.ml";;
# - : unit = ()
Findlib has been successfully loaded. Additional directives:
  #require "package";;       to load a package
  #list;;                    to list the available packages
  #camlp4o;;                 to load camlp4 (standard syntax)
  #camlp4r;;                 to load camlp4 (revised syntax)
  #predicates "p,q,...";;    to set these predicates
  Topfind.reset();;          to force that packages will be reloaded
  #thread;;                  to enable threads

- : unit = ()
/home/jeremy/.opam/4.01.0/lib/ocaml/threads: added to search path
/home/jeremy/.opam/4.01.0/lib/ocaml/unix.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ocaml/threads/threads.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc: added to search path
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc-foreign.cma: loaded
# #use "/home/jeremy/mphil-lecture/code/bind_buggy.ml";;
val libbuggy : Dl.library = <abstr>
val add : int -> int -> int = <fun>
val mul : int -> int -> int = <fun>
# 

U:**-  *ocaml-toplevel*   All L25    (Tuareg-Interactive:run +3)
Mark set
```

CTYPES BACK IN ACTION


emacs@berlioz

```
        OCaml version 4.01.0

#use "/home/jeremy/mphil-lecture/code/preliminaries_ipc.ml";;
# - : unit = ()
Findlib has been successfully loaded. Additional directives:
  #require "package";;      to load a package
  #list;;                   to list the available packages
  #camlp4o;;                to load camlp4 (standard syntax)
  #camlp4r;;                to load camlp4 (revised syntax)
  #predicates "p,q,...";;   to set these predicates
  Topfind.reset();;         to force that packages will be reloaded
  #thread;;                 to enable threads

- : unit = ()
/home/jeremy/.opam/4.01.0/lib/ocaml/threads: added to search path
/home/jeremy/.opam/4.01.0/lib/ocaml/unix.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ocaml/threads/threads.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc: added to search path
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc.cma: loaded
/home/jeremy/.opam/4.01.0/lib/ctypes_ipc/ctypes_ipc-foreign.cma: loaded
# #use "/home/jeremy/mphil-lecture/code/bind_buggy.ml";;
val libbuggy : Dl.library = <abstr>
val add : int -> int -> int = <fun>
val mul : int -> int -> int = <fun>
# add 2 3;;
- : int = 5
# mul 2 3;;
Exception: Ctypes_raw.Memory_access_error.
# 
```

```
U:**-  *ocaml-toplevel*   All L29    (Tuareg-Interactive:run +3)
Mark set
```

Itinerary

~~Background~~ / ~~using ctypes~~ / inside ctypes

Levels of type safety

algebraic data types

unsafe interface, unsafe implementation.

phantom types

safe interface, unsafe implementation.

generalized algebraic data types

safe interface, safe (and efficient!) implementation.

Other Embedded DSLs

parsing (parsec)

SQL

financial contracts

music

graphics

etc.