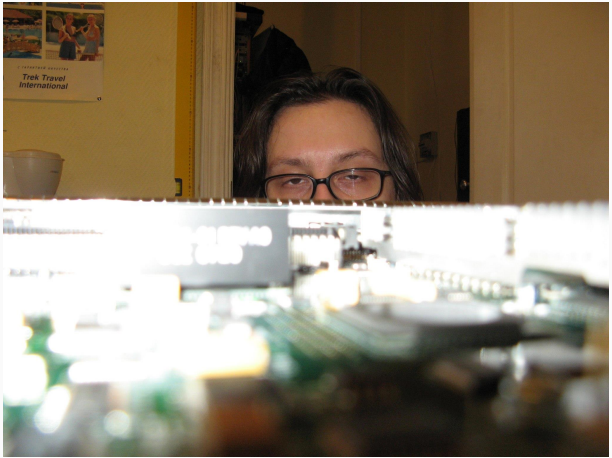
OCaml

Знакомство

Павел Аргентов

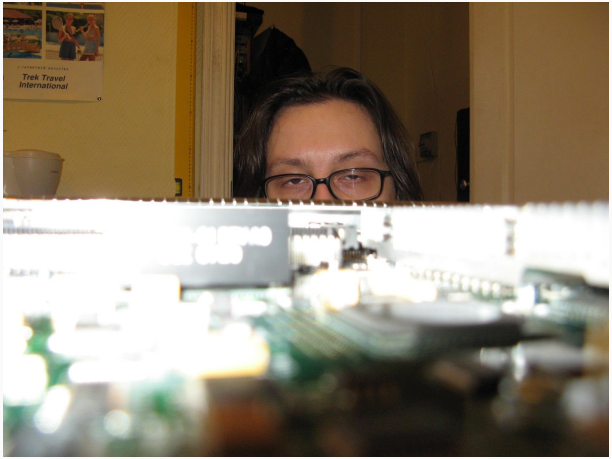
18 апреля 2020 г.

$ whois Pavel Argentov



- Hello, IT: 3 г.
- Telecom, ISP: 10 л.
- Webdev, backend
  2012 — …

$ whois Pavel Argentov



где-то в 2005

$ cat disclaimers

- Не буду защищать OCaml

$ cat disclaimers

- Не буду защищать OCaml
- Не буду сравнивать OCaml и Haskell

$ cat disclaimers

- Не буду защищать OCaml
- Не буду сравнивать OCaml и Haskell
- Не буду продавать ФП

$ cat disclaimers

«Я каску на стройке нашёл»

OCaml: кому это выгодно?

1. Утилиты
    - Unison
    - Mldonkey

OCaml: кому это выгодно?

1. Утилиты
    - Unison
    - Mldonkey
2. Системный софт
    - Xen
    - MirageOS

OCaml: кому это выгодно?

1. Утилиты
    - Unison
    - Mldonkey
2. Системный софт
    - Xen
    - MirageOS
3. Языки
    - Coq
    - Flow
    - ReasonML
    - BuckleScript
    - Hack

OCaml: кому это выгодно?

1. Утилиты
   - Unison
   - Mldonkey
2. Системный софт
   - Xen
   - MirageOS
3. Языки
   - Coq
   - Flow
   - ReasonML
   - BuckleScript
   - Hack

- Bloomberg L.P.
- Jane Street Capital
- Citrix Systems
- Facebook
- Docker

OCaml: кто это сделал?

- 1980s, The Origin, ML

  Robin Milner (LCF, ML), Gérard Huet (Lisps),

  Guy Cousineau (ADT, PM, CAM + ML = Caml)

OCaml: кто это сделал?

- 1980s, The Origin, ML

  Robin Milner (LCF, ML), Gérard Huet (Lisps),

  Guy Cousineau (ADT, PM, CAM + ML = Caml)

- 1987–1992, The First Implementation

  Ascander Suarez, Pierre Weis, Michel Mauny, LLM3 (Le_Lisp)

OCaml: кто это сделал?

- 1980s, The Origin, ML

  Robin Milner (LCF, ML), Gérard Huet (Lisps),

  Guy Cousineau (ADT, PM, CAM + ML = Caml)

- 1987–1992, The First Implementation

  Ascander Suarez, Pierre Weis, Michel Mauny, LLM3 (Le_Lisp)

- 1990s, Caml Light

  Xavier Leroy (BC impl'n), Damien Doligez (memory management)

OCaml: кто это сделал?

- 1996 – 2000, Objective Caml
  Didier Rémy, Jérôme Vouillon (OO subsystem)

  Jacques Garrigue (polymorphic methods, labeled/optional arguments,

  polymorphic variants)

OCaml: кто это сделал?

- 1996–2000, Objective Caml
  Didier Rémy, Jérôme Vouillon (OO subsystem)

  Jacques Garrigue (polymorphic methods, labeled/optional arguments,

  polymorphic variants)
- 2000–…, OCaml
  Xavier Leroy, BDFL

OCaml: кто это сделал?

Компилятор(-ы): версионирование

```
$ opam switch list-available
# Listing available compilers from repositories:
# Name               # Version
ocaml-base-compiler 3.07
<...>
ocaml-variants      4.11.0+trunk
ocaml-variants      4.11.0+trunk+afl
ocaml-variants      4.11.0+trunk+flambda
```

Компилятор(-ы): бэкенды

- Bytecode
- Native
  IA-32, X86-64 (AMD64), Power, SPARC, ARM, ARM64
- JS
  ocsigen/js-of-ocaml

Build systems: Dune

```
(executable
 (name rpiterm)
 (public_name rpiterm)
 (preprocess (pps lwt_ppx))
 (libraries cmdliner
            rpiterm_lib
            prometheus-app.unix)
 (flags (-safe-string))
 (ocamlopt_flags (:standard (:include ocamlopt_flags.sexp))))

(rule
 (targets ocamlopt_flags.sexp)
 (deps (:discover config/discover.exe))
 (action (run %{discover})))
```

OCaml: экосистема

Менеджеры пакетов

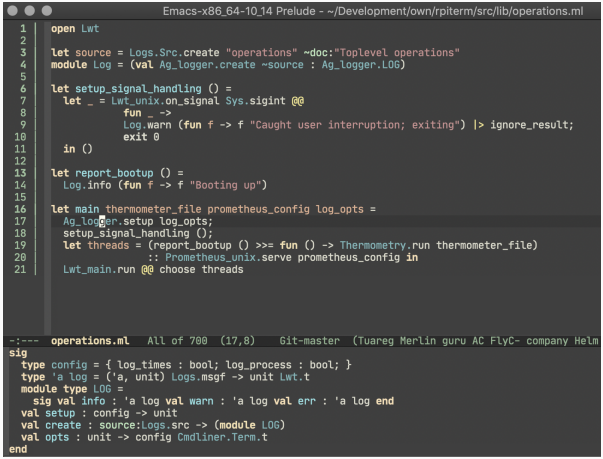
- OPAM
  - global & local compiler switching
  - dependencies isolation in local switch
  - env sandboxing

OCaml: экосистема

Менеджеры пакетов

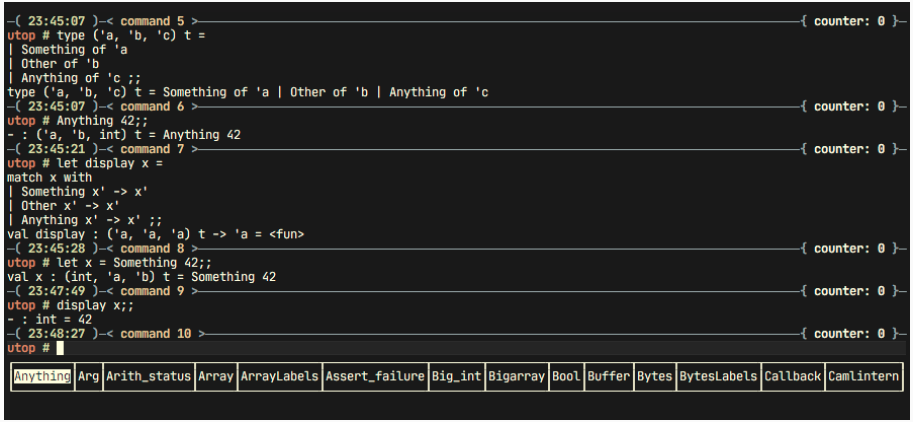
- OPAM
  - global & local compiler switching
  - dependencies isolation in local switch
  - env sandboxing
- Esy
  - package.js driven (npm style)
  - uses OPAM
  - env sandboxing
  - artifacts caching
  - Nix style

Editors integration: Merlin



OCaml: экосистема

REPL: Utop



Awesome OCAML

OCaml: напишем маленькую программку…

argent-smith/rpiterm

OCaml: напишем маленькую программку…

«Термометр» для Raspberry Pi (ARM64)

```
[INF] operations: Booting up
[INF] thermometry: Got thermometer value as 68.218
[INF] thermometry: Got thermometer value as 67.680
[INF] thermometry: Got thermometer value as 68.756
[INF] thermometry: Got thermometer value as 67.680
```

OCaml: напишем маленькую программку…

«Термометр» для Raspberry Pi (ARM64)

- Поток 1
  1. Прочитать float из файла с температурой
  2. Отдать его в Prometheus-exporter
  3. Поспать 5 секунд
  4. Повторить
- Поток 2 (Prometheus exp.)
  1. Ждать http-запрос
  2. Отдать ответ
  3. Повторить

1. Язык типов

OCaml: языки

1. Язык типов
2. Язык модулей

OCaml: языки

1. Язык типов
2. Язык модулей
3. Язык ФП

OCaml: языки

1. Язык типов
2. Язык модулей
3. Язык ФП
4. Язык ИП

OCaml: языки

1. Язык типов
2. Язык модулей
3. Язык ФП
4. Язык ИП
5. Язык ООП

OCaml: язык типов

```
1  type nonrec 'a ref = 'a ref = { mutable contents : 'a; }
2  type nonrec 'a option = None | Some of 'a
3  type nonrec int
4  type wtf = Well 'a | Nope
5  type metric_type =
6    Counter | Gauge | Summary | Histogram
7  type sample = {
8    ext    : string;
9    value  : float;
10   bucket : (LabelName.t * float) option;
11 }
12 type ft = string option -> int -> int option
```

OCaml: язык модулей

```
1  module type THERMOMETER = sig
2    val read_temperature :
3      ?t_file : string option -> unit -> float result Lwt.t
4  end
5  module Mockup : THERMOMETER = struct
6    let read_temperature ?(t_file = None) () =
7      return @@ Ok (36600.0 /. 1000.0)
8  end
9  let get_thermometer_value _file =
10   let module T = Thermometer.Linux in
11   let%lwt thermometer_value = T.read_temperature () in
12   ...
```

OCaml: язык модулей

```
1  module type X_int = sig val x : int end
2
3  module Increment (M : X_int) : X_int = struct
4    let x = M.x + 1
5  end
6
7  module Three = struct let x = 3 end
8
9  module Four = Increment(Three)
10
11 Four.x - Three.x  (* - : int = 1 *)
```

OCaml: язык модулей

```
module type LOG = sig
  val info : 'a log
  val warn : 'a log
  val err : 'a log
end
let create ~source =
  let module Src_log = (val Logs.src_log source : Logs.LOG) in
  let module Log = struct
      let info msgf = Src_log.info msgf |> return
      and warn msgf = Src_log.warn msgf |> return
      and err msgf = Src_log.err msgf |> return
    end
  in
  (module Log : LOG)
```

OCaml: язык ФП

Иммутабельность, first-class функции, карринг

```
1  open Prometheus
2
3  let namespace = "node"
4  and subsystem = "hwmon"
5
6  let chip_temperature =
7    let help = "Hardware monitor for temperature (input)"
8    and label_name = "sensor" in
9    Gauge.v_label ~help ~label_name ~namespace ~subsystem "temp_celsius"
```

LET «на выброс» и точка входа

```
let () =
  match Term.eval (operation, info) with
  | Error _ -> exit 1
  | _ -> exit 0

let setup_signal_handling () =
  let _ = Lwt_unix.on_signal Sys.sigint @@
    fun _ ->
      Log.warn (fun f -> f "Caught user interruption; exiting")
      |> ignore_result;
      exit 0
  in ()
```

Паттерны (и ppx)

```
let read_temperature ?(thermometer_file = None) () =
  match thermometer_file with
  | None -> return @@ Error No_thermometer_file
  | Some filename ->
     let open Lwt_io in
     let read_float channel =
       let%lwt str = read_line channel in
       return @@ Ok ((float_of_string str) /. 1000.0)
     in
     try%lwt
           with_file ~mode:input filename read_float
     with
     | exn -> return @@ Error exn
```

Рекурсия (и монада)

```
let rec run thermometer_file =
  Lwt_unix.sleep 5.0
  >>= (fun () -> get_thermometer_value @@ Some thermometer_file)
  >|= Prometheus.Gauge.set @@ Metrics.chip_temperature "soc_chip_temp"
  >>= fun () -> run thermometer_file
```

Мутабельность: ссылки

```
1  type 'a ref = { mutable contents : 'a }
2
3  let ref x = { contents = x }
4  let x = ref 1
5
6  let (:=) r x = r.contents <- x
7  let (!) r = r.contents
8  x := !x + 1
9
10 !x (* - : int = 2 *)
```

OCaml: язык ИП

Мутабельность: структуры (note the “;”)

```
1  type client_info =
2   { addr: Unix.Inet_addr.t;
3     port: int;
4     user: string;
5     credentials: string;
6     mutable last_heartbeat_time: Time_ns.t;
7     mutable last_heartbeat_status: string;
8   }
9
10 let register_heartbeat t hb =
11   t.last_heartbeat_time   <- hb.Heartbeat.time;
12   t.last_heartbeat_status <- hb.Heartbeat.status_message
```

Циклы

```
let quit_loop = ref false in
while not !quit_loop do
  print_string "Have you had enough yet? (y/n) ";
  let str = read_line () in
  if str.[0] = 'y' then
    quit_loop := true
done

for j = 0 to last_col do
  m3i.(j) <- inner_loop last_row 0 m1i m2 j
done
```

OCaml: язык ООП

```
1  class ['a] stack =
2    object (self)
3      val mutable list = ( [] : 'a list )  (* instance variable *)
4      method push x =                      (* push method *)
5        list <- x :: list
6      method pop =                         (* pop method *)
7        let result = List.hd list in
8        list <- List.tl list;
9        result
10     method peek =                        (* peek method *)
11       List.hd list
12     method size =                        (* size method *)
13       List.length list
14   end
```

OCaml: язык ООП

```
1 let s = new stack
2
3 s#push 3.0
4 s#pop          (* - : float = 3. *)
```

Immediate objects

```
let o =
  object
    val mutable n = 0
    method incr = n <- n + 1
    method get = n
  end

val o : < get : int; incr : unit > = <obj>
```

Thnx

@argent_smith