

OCaml

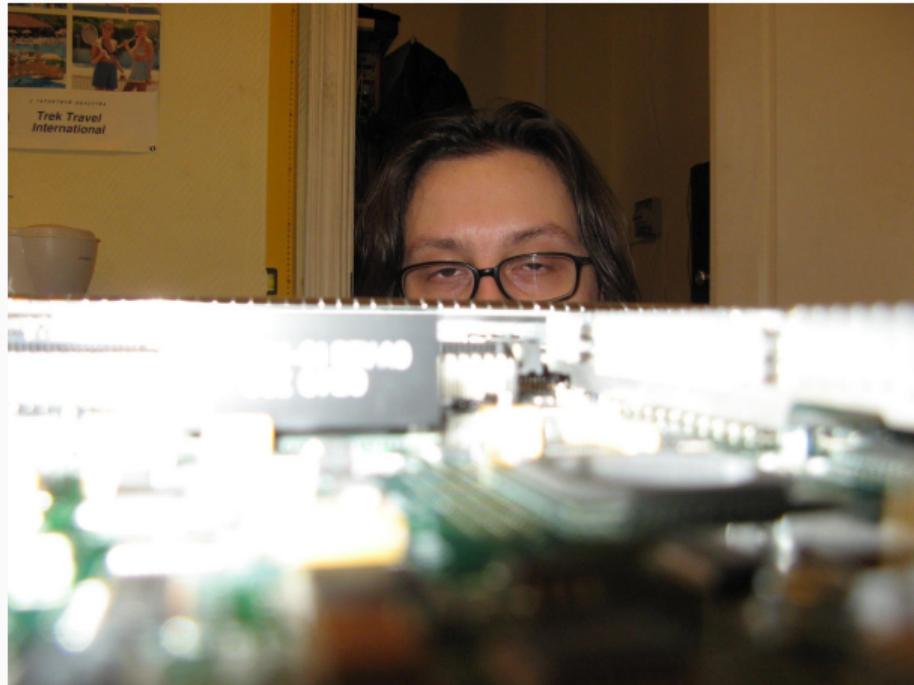
Знакомство

Павел Аргентов

4 апреля 2020 г.

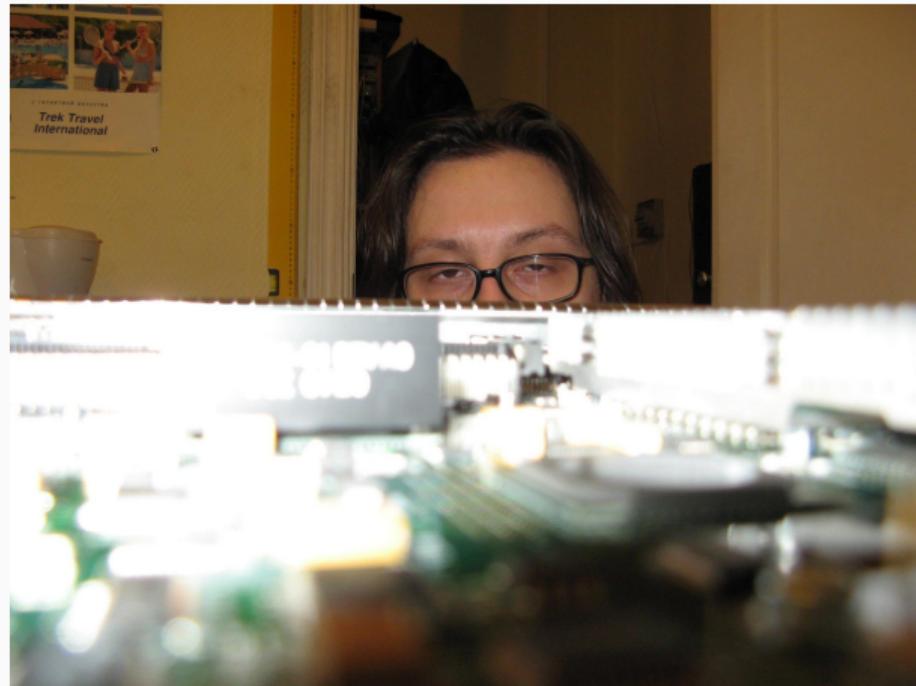


\$ whois Pavel Argentov



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

OCaml: кто это сделал?

- 1980s, The Origin, ML
Robin Milner (LCF, ML), Gérard Huet (Lisp),
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 (Lisp),
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

Компилятор(-ы): бэкенды

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

Build systems: Dune

Менеджеры пакетов

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

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

```
open Lwt
let source = Logs.Src.create "operations" ~doc:"Toplevel operations"
module Log = (val Ag_logger.create ~source : Ag_logger.LOG)
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 ()
let report_bootup () =
  Log.info (fun f -> f "Booting up")
let main thermometer_file prometheus_config log_opts =
  Ag_logger.setup log_opts;
  setup_signal_handling ();
  let threads = (report_bootup () >>= fun () -> Thermometry.run thermometer_file)
  :: Prometheus_unix.serve prometheus_config in
  Lwt_main.run @@ choose threads

```

-:--- operations.ml All of 700 (17,8) Git-master (Tuareg Merlin guru AC FlyC- company Helm
sig
type config = { log_times : bool; log_process : bool; }
type 'a log = ('a, unit) Logs.msgf -> unit Lwt.t
module type LOG =
 sig val info : 'a log val warn : 'a log val err : 'a log end
 val setup : config -> unit
 val create : source:Logs.src -> (module LOG)
 val opts : unit -> config Cmdliner.Term.t
end

OCaml: экосистема

REPL: Utop

```
( 23:45:07 )-< command 5 >-----{ counter: 0 }-
utop # type ('a, 'b, 'c) t =
| Something of 'a
| Other of 'b
| Anything of 'c;;
type ('a, 'b, 'c) t = Something of 'a | Other of 'b | Anything of 'c
-( 23:45:07 )-< command 6 >-----{ counter: 0 }-
utop # Anything 42;;
- : ('a, 'b, int) t = Anything 42
-( 23:45:21 )-< command 7 >-----{ counter: 0 }-
utop # let display x =
match x with
| Something x' -> x'
| Other x' -> x'
| Anything x' -> x' ;;
val display : ('a, 'a, 'a) t -> 'a = <fun>
-( 23:45:28 )-< command 8 >-----{ counter: 0 }-
utop # let x = Something 42;;
val x : (int, 'a, 'b) t = Something 42
-( 23:47:49 )-< command 9 >-----{ counter: 0 }-
utop # display x;;
- : int = 42
-( 23:48:27 )-< command 10 >-----{ counter: 0 }-
utop #
```

Anything	Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Bool	Buffer	Bytes	BytesLabels	Callback	Camlintern
----------	-----	--------------	-------	-------------	----------------	---------	----------	------	--------	-------	-------------	----------	------------

Awesome OCAML



OCaml: напишем маленькую программку...

argent-smith/rpiterm



OCaml: напишем маленькую программку...

«Термометр» для Raspberry Pi (ARM64)

«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

1. Язык типов
2. Язык ФП

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

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

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

OCaml: язык типов

OCaml: язык модулей