Peter the Great
Saint-Petersburg Polytechnic University

# Ocaml: многопоточность, сборка и экосистема

Презентацию подготовили:
- Дрекалов Никита
- Соколов Артём

POLYTECH

12/12/2025

# План презентации

- Многопоточность
  - Системные потоки и доменная модель
- Механизмы синхронизации
  - Atomic, Mutex и CV
- Обработка исключений
  - Определение и обработка исключений
- Управление пакетами с Opam
  - Основные команды и файл конфигурации
- Система сборки Dune
  - Структура проекта, конфиг файлы
- Тестирование в OCaml
  - Alcotest, OUnit, QCheck
- Деплой и контейнеризация
  - Dockerfile, профили сборки

# Многопоточность

# Системные потоки

- В OCaml можно создавать системные потоки через модуль Thread.
- Но из-за Global Runtime Lock (GRL, аналог GIL в Python) только один поток OCaml-кода выполняется одновременно.

```
let thread_fun () =
  for i = 1 to 5 do
    print_endline ("Thread: " ^ string_of_int i);
    Thread.delay 0.5
  done

let () =
  let t = Thread.create thread_fun () in
  thread_fun ();
  Thread.join t
```

# Доменная модель (domains)

- Начиная с Ocaml 5, появилась новая модель параллелизма: Domains, которые позволяют выполнять Ocaml-код параллельно на разных ядрах, избегать GRL и использовать Thread-пулы («effects»)

```
open Domainslib

let () =
  let pool = Task.setup_pool ~num_additional_domains:3 () in
  let r =
    Task.parallel_for_reduce pool ~start:0 ~finish:1_000_000
      ~body:(fun _ -> 1)
      ~reduce:(+) ~init:0
  in
  Printf.printf "Result = %d\n" r;
  Task.teardown_pool pool
```

# Механизмы синхронизации

# Lock-free структуры

В OCaml есть примитивы:

- Atomic.t — атомарная ячейка
- Atomic.compare_and_set
- Atomic.get / Atomic.set

```
let counter = Atomic.make 0

let increment () =
  let rec loop () =
    let old = Atomic.get counter in
    if Atomic.compare_and_set counter old (old + 1)
    then ()
    else loop ()
  in loop ()
```

# Mutex

В OCaml есть классический механизм синхронизации с блокировкой потоков через мьютексы

```
let m = Mutex.create ()

let critical () =
  Mutex.lock m;
  print_endline "critical";
  Mutex.unlock m
```

# Condition variables

Механизм синхронизации, позволяющий одному потоку ждать, пока другой поток сообщит, что произошло событие

```
(* Общие ресурсы *)
let mutex = Mutex.create ()
let cond = Condition.create ()
let queue = Queue.create ()

(* Производитель: добавляет задачи *)
let producer () =
  for i = 1 to 5 do
    Mutex.lock mutex;
    Queue.push ("task " ^ string_of_int i) queue;
    (* Сообщаем, что очередь не пустая *)
    Condition.signal cond;
    Mutex.unlock mutex;
    Thread.delay 1.0  (* имитируем работу *)
  done
```

```
(* Потребитель: ждёт задач *)
let consumer () =
  while true do
    Mutex.lock mutex;
    (* Пока очередь пуста — ждём *)
    while Queue.is_empty queue do
      Condition.wait cond mutex
    done;
    (* Когда проснулся — mutex снова захвачен *)
    let task = Queue.pop queue in
    Mutex.unlock mutex;
    Printf.printf "Consumed: %s\n%!" task
  done


    let () =
      let c = Thread.create consumer () in
      let p = Thread.create producer () in
      Thread.join c;
      Thread.join p
```

POLYTECH

# Обработка исключений

# Определение и генерация исключений

- Ocaml поддерживает классическую модель исключений, похожую на ML и немного на C++

```
exception My_error of string

let do_stuff x =
  if x < 0 then
    raise (My_error "negative!")
  else
    x + 1

try
  do_stuff (-1)
with
| My_error msg -> Printf.printf "Error: %s\n" msg
| _ -> print_endline "Unknown error"
```

# Управление пакетами с Opam

# Основные команды opam

- opam – менеджер пакетов для OCaml

```
opam init

# Установка пакетов
opam install core          # установка библиотеки Core
opam install dune merlin   # установка инструментов сборки

# Поиск пакетов
opam list                  # список установленных пакетов
opam search json           # поиск пакетов по имени

# Обновление
opam update                # обновление информации о пакетах
opam upgrade               # обновление установленных пакетов
```

# Инициализация и создание `switch`

- `switch` — изолированное окружение с конкретной версией компилятора и набором пакетов

```
opam init
eval $(opam env)

opam switch create 4.14.0 ocaml-base-compiler.4.14.0
eval $(opam env)

opam install . --deps-only
```

# Файл конфигурации opam

```
opam-version: "2.0"
name: "myproject"
version: "0.1.0"

build: [
  ["dune" "build" "-p" name "-j" jobs]
]

depends: [
  "ocaml" {>= "4.08.0"}
  "dune" {>= "2.0.0"}
  "cohttp-async"
]
```

# Система сборки Dune

# Dune и структура проекта

- Dune – современная система сборки для OCaml
- Проект организуется в директории bin, lib, test
- Каждая директория содержит файл dune с правилами сборки

```
myapp/
  dune-project
  bin/
    dune
    main.ml
  lib/
    dune
    myapp.ml
  tests/
    dune
    tests.ml
```

# Конфигурационные файлы Dune

tests/dune

```
(tests
 (names test_my_lib)
 (libraries core my_lib ounit2)
 (preprocess (pps ppx_jane)))
```

main/dune

```
(executable
 (name main)
 (public_name myproject)
 (libraries core my_lib)
 (preprocess (pps ppx_jane)))
```

lib/dune

```
(library
 (name my_lib)
 (public_name myproject.lib)
 (libraries core yojson)
 (preprocess (pps ppx_jane)))
```

# Конфигурационные файлы Dune

```
(lang dune 3.8)
(name myproject)
(package
 (name myproject)
 (synopsis "A cool OCaml project")
 (description "A longer description of the project")
 (depends ocaml dune)
 (authors "Your Name")
 (license MIT))
```

# Тестирование в OCaml

# Alcotest

```
open Alcotest


let test_add () =
  check int "same int" 3 (1 + 2)


let () =
  run "myapp-test" [
    "math", [
      test_case "add" `Quick test_add
    ];
  ]
```

# OUnit2

```
open OUnit2


let test_add _ = assert_equal 3 (1+2)


let () = run_test_tt_main (
  "suite" >:: [
    "add" >:: test_add;
  ])
```

# QCheck

```
open QCheck

let add_is_commutative =
  Test.make ~count:10_000
    (pair int int)
    (fun (x, y) -> x + y = y + x)
```

# Деплой и контейнеризация

# Dockerfile для OCaml приложения

```
FROM ocaml/opam:alpine-ocaml-5.1 AS build

WORKDIR /app
COPY my_project.opam dune-project ./
RUN opam install . --deps-only -y

COPY . .
RUN opam exec -- dune build --profile=release

FROM alpine:latest

RUN apk add --no-cache libev gmp

COPY --from=build /app/_build/default/bin/main.exe /usr/local/bin/myapp


CMD ["/usr/local/bin/myapp"]
```

# Профили сборки

```
(profile release)
(env
 (release
  (flags (:standard -O3 -unbox-closures))
  (ocamlopt_flags (:standard -O3))))

(profile dev)
(env
 (dev
  (flags (:standard -g -warn-error -A))
  (ocamlopt_flags (:standard -g))))

(profile profiling)
(env
 (profiling
  (flags (:standard -p -g))
  (ocamlopt_flags (:standard -p -g))))
```

# Спасибо за внимание!