OCaml简介

2015-03-27

# 安装与配置

OCaml的安装项有

* OCaml
* 相关lib
* 包管理工具OPAM
* 更好的交互式工具utop
* 更好的基础库扩展Core

## 安装OCaml以及相关lib

Ubuntu

apt-get install ocaml ocaml-nox ocaml-native-compilers ocaml-doc ocaml-findlib oasis libpcre-ocaml-dev

Fedora

yum install ocaml ocaml-foo-devel ocaml-camlp4-devel ocaml-ocamldoc ocaml-findlib-devel ocaml-extlib-devel ocaml-calendar-devel

## 安装OPAM

Ubuntu

apt-get install opam

Fedora需要先下载[RPM包](http://software.opensuse.org/download.html?project=home%3Aocaml&package=opam)，然后使用rpm安装，类似于

rpm -ivh opam.rpm

安装完毕后需要使用普通用户进行配置

opam init

opam switch 4.01.0

eval `opam config env`

## 安装utop

opam install utop

## 安装与配置Core

opam install core

opam install async yojson core\_extended core\_bench cohttp async\_graphics cryptokit menhir

然后编辑文件~/.ocamlinit，添加内容

#use "topfind";;

#thread;;

#camlp4o;;

#require "core.top";;

#require "core.syntax";;

#require “async”;;

# OCaml语言简介

OCaml是一种函数式编程语言，并且兼有命令式、面向对象的特点，同时有很高的性能。OCaml已经在工业界有成熟的应用。

## 基本概念

一下简单针对注释、数值运算、基本类型（Tuple/List/Option/Record/Variant）、变量和函数进行了示例。注意每个语句最后的;;只是为了告诉交互式工具可以开始求值了，并不是必需的。

(\* comments \*)

(\* integer operation \*)

1 + 2;;

(\* float number operation \*)

1. +. 2.;;

(\* tuple \*)

(1, “a”, 99.9);;

(\* list \*)

[1; 2; 3;];;

(\* option list \*)

[Some 1; None];;

(\* record \*)

type r = {a: int; b: string}

{a = 1; b = “red”};

(\* variant \*)

type v = | Int of int \* int | String of string

[Int(1, 2); String(“center”)];;

(\* variable binding \*)

let x = 1;;

(\* function binding \*)

let add x y = x + y;;

(\* recursive function and pattern matching \*)

let rec has\_none x =

match x with

| [] -> false

| None::\_ -> true

| \_::rest -> has\_none rest

;;

(\* mutually recursive function \*)

let rec is\_even x =

if x = 0 then true else is\_odd (x - 1)

and is\_odd x =

if x = 0 then false else is\_even (x - 1);;

## 代码组织

OCaml代码以模块的方式组织，一个模块由模块接口（.mli文件）和模块实现（.ml）文件组成，且.mli和.ml文件的命名需要完全一致，且全部由小写字母和下划线组成。这样建立的模块对应的模块名称与文件名相同，但是首字母需要大写。

下面是一个简单示例，用来在一个字符串列表中判断一个字符串是否存在

(\* File find\_str.mli \*)

type t

val find : string list -> string -> bool

(\* File find\_str.ml \*)

open Core.Std

type t = string list

let find strlist str =

match List.find ~f:(fun x -> x = str) strlist with

| None -> false

| Some \_ -> true

实际上，在上边的.ml文件中，已经使用了其他的模块。如果我们需要在另一个文件中引用模块Find\_str，可以使用

open Find\_str

find [“a”; “b”] “a”

## 工程管理

//todo

## 多进程管理

/todo