OCamlbuild par l'exemple

Jacques-Pascal Deplaix - OUPS

Introduction

OCamlbuild is a build-system for OCaml among others (like ocp-build, obuild, omake, OMakefile, OCamlMakefile, ...)

- Pros
  - Distributed with OCaml
  - Configuration in OCaml
  - Simple (example: used in Real World OCaml)
- Cons
  - Parallel builds
  - Unmaintened for years

How does it works ?

```
|– _build
|– _tags
|– a.ml
|– b.ml
|_ myocamlbuild.ml
```

```
$ ocamlbuild a.native
```

_tags: a simple config file that contains relations between files and tags
_build: the build directory
myocamlbuild.ml: plugin file (in OCaml)

What is a tag ?

```
<*.ml>: package(js_of_ocaml.syntax), syntax(camlp4o), debug
true: use_menhir
"a.js": opt(2)
```

Example: the js_of_ocaml plugin

```
open Ocamlbuild_plugin

let init () =
  let dep = "%.byte" in
  let prod = "%.js" in
  let f env _ =
    let dep = env dep in
    let prod = env prod in
    let link_opts = link_opts prod in
    let tags = tags_of_pathname dep ++ "js_of_ocaml" in
    Cmd (S [A "js_of_ocaml"; T tags; S link_opts; P dep; A "-o"; Px prod])
  in
  rule "js_of_ocaml: .byte -> .js" ~dep ~prod f;
  flag ["js_of_ocaml"; "debug"] (S [A "-pretty"; A "-debuginfo"; A "-noinline"]);
  pflag ["js_of_ocaml"] "opt" (fun n -> S [A "-opt"; A n])
```

Mise en avant du dynamisme

Dispatching et utilisation

```
open Ocamlbuild_plugin

module M = Ocamlbuild_eliom.Make(struct
  let client_dir = "client"
  let server_dir = "server"
  let type_dir = "type"
end)

let () =
  dispatch
    (fun hook ->
       dispatch_default hook;
       M.dispatcher hook;
       match hook with
         | After_options ->
             let f = function
               | "src/client/cumulus.byte" -> "src/client/cumulus.js"
               | x -> x
             in
             Options.targets := List.map f !Options.targets
         | _ -> ()
    )
```

Questions ?

« C'est pas faux ! »