



# Preface

*This is a book about hacking in ocaml. It's assumed that you already understand the underlying theory. Happy hacking Most parts are filled with code blocks, I will add some comments in the future. Still a book in progress. Don't distribute it.*

☺



# Acknowledgements

write  
later



# Contents

<b>Preface</b>	<b>3</b>
<b>Acknowledgements</b>	<b>5</b>
<b>1 Tool Chain</b>	<b>15</b>
1.1 Ocamlbuild . . . . .	16
multiple directories . . . . .	19
grouping targets . . . . .	20
With lex yacc, ocamlfind . . . . .	21
1.1.1 Principles . . . . .	22
1.1.2 Write plugin . . . . .	23
Samples . . . . .	26
Mixing with C stubs . . . . .	28
1.1.3 OCamlbuild in the toplevel . . . . .	29
1.1.4 Building interface files . . . . .	29
Interaction with Git . . . . .	29
1.2 Godi,otags . . . . .	30
1.2.1 CheatSheet . . . . .	30
1.3 Ocamlfind . . . . .	31
1.3.1 CheatSheet . . . . .	31
1.3.2 META file . . . . .	31
1.4 toplevel . . . . .	32
1.4.1 directives . . . . .	32

1.4.2	Module toplevel . . . . .	32
1.4.3	Env . . . . .	33
1.5	OcamlDoc . . . . .	35
1.6	ocamlmktop . . . . .	38
1.7	Standard OCaml Develop Tools . . . . .	39
1.8	ocamlmktop . . . . .	41
1.9	Git . . . . .	43
<b>2</b>	<b>Lexing</b>	<b>45</b>
2.1	Lexing . . . . .	46
2.1.1	Ulex interface . . . . .	47
2.2	Ocamllex . . . . .	55
<b>3</b>	<b>Parsing</b>	<b>59</b>
3.1	OcamlYacc . . . . .	60
3.2	MENHIR Related . . . . .	74
<b>4</b>	<b>Camlp4</b>	<b>77</b>
4.1	Predicate Parser . . . . .	78
4.2	Basic Structure . . . . .	79
4.2.1	Experimentation Environment . . . . .	79
4.2.2	Camlp4 Modules . . . . .	79
4.2.3	Camlp4 Make Toplevel . . . . .	80
4.2.4	Command Options . . . . .	83
4.2.5	Module Components . . . . .	84
4.2.6	Simple Experiment . . . . .	85
4.3	Camlp4 SourceCode Exploration . . . . .	86
4.3.1	Camlp4 PreCast . . . . .	86
4.3.2	OCamlInitSyntax . . . . .	88
4.3.3	Camlp4.Sig . . . . .	93
4.3.4	Camlp4.Struct.Camlp4Ast.mlast . . . . .	93
4.3.5	AstFilters . . . . .	94

4.3.6	Camlp4.Register . . . . .	94
4.3.7	Camlp4Ast . . . . .	99
4.3.8	TestFile . . . . .	127
4.4	Extensible Parser . . . . .	128
4.4.1	Examples . . . . .	128
4.4.2	Mechanism . . . . .	129
4.4.3	Parsing OCaml using Camlp4 . . . . .	134
	Fully Utilize Camlp4 Parser and Printers . . . . .	134
	Otags Mini . . . . .	134
	Parsing Json AST . . . . .	136
4.5	STREAM PARSER . . . . .	138
4.6	Grammar . . . . .	139
4.6.1	LEVEL . . . . .	141
4.6.2	Grammar Modification . . . . .	141
	Example: Expr Parse Tree . . . . .	144
4.7	QuasiQuotations . . . . .	149
4.7.1	Quotation Introduction . . . . .	150
4.7.2	Quotation Expander . . . . .	151
	Lambda Example . . . . .	155
4.8	Ast Transformation . . . . .	159
4.13	Examples . . . . .	182
4.13.1	Pa_python . . . . .	182
4.13.2	Pa_list . . . . .	187
4.13.3	Pa_abstract . . . . .	189
4.13.4	Pa_apply . . . . .	190
4.13.5	Pa_ctyp . . . . .	190
4.13.6	Pa_exception_wrapper . . . . .	191
4.13.7	Pa_exception_tracer . . . . .	194
4.13.8	Pa_freevars . . . . .	195
4.13.9	Pa_freevars_filter . . . . .	195
4.13.10	Pa_global_handler . . . . .	195



4.13.11 Pa_holes . . . . .	195
4.13.12 Pa_minimm . . . . .	195
4.13.13 Pa_plus . . . . .	195
4.13.14 Pa_zero . . . . .	195
4.13.15 Pa_printer . . . . .	195
4.13.16 Parse_arith . . . . .	196
4.13.17 Pa_estring . . . . .	196
4.13.18 Pa_holes . . . . .	205
4.14 Useful links . . . . .	206
4.15 Camlp4 CheatSheet . . . . .	207
4.15.1 Camlp4 Transform Syntax . . . . .	207
Example: semi opaque . . . . .	207
4.15.2 Parsing . . . . .	208
4.11 Revised syntax . . . . .	168
<b>5 Libraries</b>	<b>175</b>
5.1 Format . . . . .	210
5.1.1 Indentation Rules . . . . .	211
5.1.2 Boxes . . . . .	211
5.1.3 Directives . . . . .	213
5.1.4 Example: Print . . . . .	213
5.2 ocamlgraph . . . . .	214
5.2.1 Undirected graph . . . . .	215
5.2.2 Example Visualize Dependency . . . . .	216
5.3 batteries . . . . .	220
syntax extension . . . . .	220
5.3.1 Dev . . . . .	220
5.3.2 BOLT . . . . .	221
5.4 Mikmatch . . . . .	222
5.5 pa-do . . . . .	235
5.6 num . . . . .	236

5.7	caml-inspect . . . . .	237
5.8	pa-monad . . . . .	242
5.9	bigarray . . . . .	246
5.10	sexplib . . . . .	247
5.11	bin-prot . . . . .	250
5.12	fieldslib . . . . .	251
5.13	variantslib . . . . .	252
5.14	delimited continuations . . . . .	253
5.15	shcaml . . . . .	259
5.16	deriving . . . . .	260
5.17	Modules . . . . .	261
<b>6</b>	<b>Runtime</b>	<b>265</b>
6.1	ocamlrun . . . . .	266
6.2	FFI . . . . .	268
6.2.1	Data representation . . . . .	269
6.2.2	Caveats . . . . .	279
<b>7</b>	<b>GC</b>	<b>281</b>
<b>8</b>	<b>Object-oriented</b>	<b>289</b>
8.1	Simple Object Concepts . . . . .	290
8.2	Modules vs Objects . . . . .	294
8.3	More about class . . . . .	295
<b>9</b>	<b>Language Features</b>	<b>297</b>
9.1	Stream Expression . . . . .	298
9.2	GADT . . . . .	302
9.3	First Class Module . . . . .	303
9.4	Pahantom Types . . . . .	307
9.4.1	Useful links . . . . .	314
9.5	Positive types . . . . .	315

9.6	Private Types . . . . .	316
9.7	Subtyping . . . . .	318
9.8	Explicit Nameing Of Type Variables . . . . .	319
9.9	The module Language . . . . .	320
<b>10</b>	<b>subtle bugs</b>	<b>321</b>
10.1	Reload duplicate modules . . . . .	322
10.2	debug . . . . .	324
10.3	Debug Cheat Sheet . . . . .	325
<b>11</b>	<b>Interoperating With C</b>	<b>327</b>
<b>12</b>	<b>Pearls</b>	<b>329</b>
12.1	Write Printf-Like Function With Ksprintf . . . . .	330
12.2	Optimization . . . . .	330
12.3	Weak Hashtbl . . . . .	330
12.4	Bitmatch . . . . .	330
12.5	Interesting Notes . . . . .	331
12.6	Polymorphic Variant . . . . .	332
<b>13</b>	<b>Compiler</b>	<b>337</b>
13.1	module Printtyp . . . . .	338
<b>14</b>	<b>XX</b>	<b>339</b>
13.0.1	tricks . . . . .	338
13.0.2	ocaml blogs . . . . .	342
<b>14</b>	<b>Topics</b>	<b>343</b>
14.1	First Order Unification . . . . .	344
14.1.1	First-order terms . . . . .	344
14.1.2	Substitution . . . . .	345
14.1.3	Unification in Various areas . . . . .	345
14.1.4	Occurs check . . . . .	346

14.1.5 Unification Examples . . . . .	346
14.1.6 Algorithm . . . . .	346
14.2 LLVM . . . . .	350

## Todo list

write later . . . . .	5
mlpack file . . . . .	18
Glob Patterns . . . . .	21
parser-help to coordinate menhir and ulex . . . . .	54
predicate parsing stuff . . . . .	78
Should be re-written later . . . . .	281
Write later . . . . .	296
read ml 2011 workshop paper . . . . .	302
Read the slides by Jacques Garrigue . . . . .	303
write later with subtyping . . . . .	315
write later . . . . .	320
polymorphic comparison . . . . .	322
Write later . . . . .	327



# Chapter 1

## Tool Chain



## Chapter 2

### Lexing





# Chapter 3

## Parsing

# Chapter 4

## Camlp4

Camlp4 stands for Preprocess-Pretty-Printer for `OCaml`, it's extremely powerful and hard to grasp as well. It is a source-to-source level translation tool.



# Chapter 5

## Libraries

## 5.17 Modules



## Chapter 6

### Runtime





# Chapter 7

GC

Should  
be re-  
written  
later



## Chapter 8

# Object-oriented

Write  
later



## Chapter 9

# Language Features

write  
later

## 9.9 The module Language

# Chapter 10

## subtle bugs





# Chapter 11

## Interoperating With C

Write  
later

---



# Chapter 12

## Pearls



# Chapter 13

## Compiler

If you have *Godi* installed, then you have already compiler libs installed.

---

```
1 ls 'ocamlfind ocamlc -where'../compiler-lib
```

---

You can play with compiler lib in the toplevel

---

```
1 #directory '+../compiler-lib' ;; (** for cmi file*)
2 #load 'toplevellib.cma';;
```

---

Listing 58: Compiler lib in toplevel

## 13.1 module Printtyp

This module mainly export some printing functions for ocaml ast.

---

```
1 val longident: formatter -> Longident.t -> unit
2 val ident: formatter -> Ident.t -> unit
3 val type_expr: formatter -> type_expr -> unit
4 val modtype: formatter -> module_type -> unit
5 val signature: formatter -> signature -> unit
6 val tree_of_modtype_declaration: Ident.t -> modtype_declaration -> out_sig_item
7 val modtype_declaration: Ident.t -> formatter -> modtype_declaration -> unit
8 val class_type: formatter -> class_type -> unit
```

---

Listing 59: Printer in compiler lib

You can use this library to process *cmi* files, The structure of *cmi* file is organized as follows

---

```
1 let ic = open_in_bin filename in
2 let magic_len = String.length (Config.cmi_magic_number) in
3 let buffer = String.create magic_len in
4 really_input ic buffer 0 magic_len ;
5 let (name, (sign:Types.signature)) = input_value ic in
6 let (crcs : (string * Digest.t) list) = input_value ic in
7 let (flags : flags list) = input_value ic in
8 close_in ic ;
```

---

Listing 60: Structure of cmi file

# Chapter 14

XX



# Chapter 14

## Topics