Slides to Accompany *Programming Languages and Methodologies*

R. J. Schalkoff

Chapter 11, Part 2-Addition: `ocaml` Compilation 2015

Executing a Script

The ocaml command starts the toplevel system for Objective Caml. This is the interactive read-eval-print loop. It does not need to be interactive, since it allows specification of a script file via:

```
ocaml  [ script-file ]
```

A Script (ocaml source)

```
(** file: oddeven2015.ml; rev. 10-20-2015 *)
(** to show interpretation vs. 'real' compilation *)
(* special comments to illustrate ocamldoc *)

let rec even n =
if (n==0) then true
          else odd (n-1)
and (* here's the mutual recursion *)
odd m =
if (m==0) then false
          else even (m-1);;

(** some time-consuming evaluations; hopefully reasonable *)

Printf.printf "\neven(1000000000) is %b \n" (even 1000000000);;

Printf.printf "\neven(1000000001) is %b \n" (even 1000000001);;
```

Running and Timing the Script

First, do a 'man' on 'time'.

Example (with timing) – 2 runs

```
$time ocaml oddeven2015.ml

even(1000000000) is true

even(1000000001) is false

real 1m42.446s
user 1m42.388s
sys 0m0.090s
$time ocaml oddeven2015.ml

even(1000000000) is true

even(1000000001) is false
```

```
real 1m42.366s
user 1m42.309s
sys 0m0.089s
```

Compilation (ocamlc)

The Objective Caml bytecode compiler ocamlc compiles Caml source files to bytecode object files and links these object files to produce standalone bytecode executable files. These executable files are then run by the bytecode interpreter ocamlrun.

Compilation using ocamlc:

```
$ocamlc -o oddeven2015.ocamlc
```

It is noteworthy that under Unix (including linux), the first line of the compiled and linked file contains the location of the `ocamlrun` interpreter.

You can use vi (or any editor w/ a binary capability) to see it.

This means the file can be executed directly (without using `ocamlrun`).

In this way, you can distribute compiled and linked files.

Running (and timing) the Compiled Version

```
$time ./oddeven2015.ocamlc

even(1000000000) is true

even(1000000001) is false

real 1m42.403s
user 1m42.336s
sys 0m0.094s
```

Native Compilation (ocamlopt)

The Objective Caml high-performance native-code compiler `ocamlopt` compiles Caml source files to native code object files and links these object files to produce standalone executables.

Notes:

1. You cannot mix native-code object files produced by `ocamlopt` with bytecode object files produced by `ocamlc`: a program must be compiled entirely with `ocamlopt` or entirely with `ocamlc`.
2. Native-code object files produced by `ocamlopt` cannot be loaded in the toplevel `ocaml` system.

Use (note there are MANY switches/options):

```
$ocamlopt -o oddeven2015.ocamlopt oddeven2015.ml
```

Running and Timing the Native Version

```
$time ./oddeven2015.ocamlopt

even(1000000000) is true

even(1000000001) is false

real 0m3.094s
user 0m3.091s
sys 0m0.005s
$time ./oddeven2015.ocamlopt

even(1000000000) is true

even(1000000001) is false

real 0m3.097s
user 0m3.093s
sys 0m0.005s
```

Summary

Note: We need to be careful when developing serious 'benchmarks'. These are just an example on my Samsumg Core i3 under Ubuntu linux.

Typical timings:

- `ocaml` (script): 1m42.403s
- `ocamlc`: (same)–Why?
- `ocamlopt` (native): 0m3.1s
- Ratio (`ocamlrun/ocamlopt`): 33