It Should Just Work™

Harmonious coexisting and relocatable compilers and runtimes

Problem #1

```
dra@eoan:~$ which camlp4
/usr/bin/camlp4
dra@eoan:~$ camlp4 -v
Fatal error: cannot load shared library dllunix
Reason: /home/dra/.opam/4.08.1/lib/ocaml/stublibs/dllunix.so:
undefined symbol: caml_sigmask_hook
```

“Solution” #1

CAML_LD_LIBRARY_PATH was set for an opam switch

```
dra@eoan:~$ CAML_LD_LIBRARY_PATH= camlp4 -v
Camlp4 version 4.05.0
```

Problem #2

```
dra@eoan:~/project$ opam switch create . ocaml-base-compiler.4.09.0
dra@eoan:~/project$ ocamlc -o hello hello.ml
dra@eoan:~/project$ ./hello
Hello, world!
dra@eoan:~/project$ cd ..
dra@eoan:~$ mv project hello
dra@eoan:~$ hello/hello
bash: hello/hello: /home/dra/project/_opam/bin/ocamlrun: bad
interpreter: No such file or directory
```

“Solution” #2

Use ocamlopt, ocamlc -custom or ocamlc -output-complete-obj

Problem #3

```
dra@eoan:~/hello$ ocamlc -o hello -custom hello.ml
Command 'ocamlc' not found, but can be installed with:
sudo apt install ocaml-nox

dra@eoan:~/hello$ _opam/bin/ocamlc -o hello -custom hello.ml
File "command line", line 1:
Error: Unbound module Stdlib
```

"Solution" #3

```
dra@eoan:~/hello$ _opam/bin/ocamlc -o hello -custom hello.ml \
                                   -I _opam/lib/ocaml
```

The same problem can happen on Windows where `OCAMLLIB` is commonly set

Why does this matter?

- Being able to duplicate the compiler will allow it to be shared between opam switches more safely than simply by setting `OCAMLLIB`
- These issues catch out both beginners and more experienced users (especially in difficult-to-debug situations like CI) – the error which is seen doesn't readily relate to the problem
- Some of the workarounds aren't ideal (`-output-complete-obj`, etc.)

Goals

1. A bytecode executable with a `#!` header should either find a *correct* runtime, or fail with a reasonable description of the problem.
2. It should be possible to have multiple runtimes in `PATH`.
3. Dynamic libraries containing C primitives should be loaded for a matching version (and configuration) of the runtime.
4. It should be possible to move an installed compiler to a different location without setting environment variables.
5. The value of `OCAMLLIB` should ideally not cause a compiler to cease working just because it points to a different version of OCaml.

The Runtime ID

- Derive an 8 character ID for a given configuration of OCaml
- Configuration includes:
  - Version
  - Bytecode magic number
  - Any relevant aspects of the runtime (`-no-naked-pointers`, etc.)
- ID is the first 8 characters of the MD5
- Could encode the runtime variant as well?

Dynamic C primitives (C stubs)

- Include the runtime ID in the name of the stub library
- So `dllunix.so` becomes `dllunix-`*`RuntimeID`*`.so`
- Add a new option to `ocamlmklib` to name stub libraries this way
- At release, warning if the new option isn't used
- Three releases later, new behaviour only
- Opt-in from build systems (update `ocamlbuild` & `dune` at first release)

Runtime

- The same trick can be used for the runtime itself
- Rename `ocamlrun` to `ocamlrun-`*`RuntimeID`*
- Install a symlink for `ocamlrun` for compatibility (for humans!)

Environment variables

- Both runtime and compiler use `OCAMLLIB-`*`RuntimeID`* if defined
- Search `CAML_LD_LIBRARY_PATH-`*`RuntimeID`* before `CAML_LD_LIBRARY_PATH`
- If `OCAMLLIB` is defined, check the magic number of `stdlib.cma` and ignore `OCAMLLIB` if it doesn't match??
- Could rename `stdlib.cma` to `stdlib-`*`SystemID`*`.cma`??
- Could display an informative error (i.e. suggest clearing `OCAMLLIB`)??

camlheader

- The bytecode header now becomes a small script (NB It's **already** a script sometimes):

```
#!/bin/sh -e
i=absolute-path-to-ocamlrun-RuntimeMD5
if ! test -f "$i" || ! test -x "$i" ; then
  i=ocamlrun-RuntimeMD5
fi
exec "$i" "$@"
# Could run without set -e and have an error message here
```

- C headers updated to use the same logic
- Windows would now use the same logic (i.e. absolute path first)

`--enable-relative-libdir`

- New `configure` option
- Embed relative location of `LIBDIR` from `BINDIR` instead of `LIBDIR`
- Calculate the absolute path on each invocation of the driver
- Continue to display an absolute path for `ocamlc -where`
- Display absolute path and relative path in `ocamlc -config`

Summary

- Introduce a runtime ID, which is easily computed and unique across OCaml CPU architecture and configuration
- Make the name of the runtime sufficiently distinct that searching for it works in bytecode executable headers
- Use the same trick for stub libraries so that `CAML_LD_LIBRARY_PATH` can safely include directories for multiple versions of OCaml
- Introduce some level of hardening for invalid `OCAMLLIB` values
- Include a new configuration option to compute the location of the standard library relative to `ocamlc`/`ocamlopt`

It should then

Just Work™