opam on Windows

Getting over the line


THE PROBLEM
IS CHOICE

Picking an OCaml

- 4 officially supported OCamls to consider on Windows:
  - **MSVC** PE + Microsoft C runtime; built with Microsoft's C compiler
  - **mingw-w64** PE + Microsoft C runtime; built with mingw-w64 ported GCC
  - **Cygwin** PE + Cygwin POSIX runtime; built with stock GCC
  - **LXSS (WSL)** ELF + glibc; built with stock GCC
- The PE compilers exist in both i686 and x86_64 variants

Picking an environment

- The *environment* provides external commands and libraries
- In opam now, this is simply what is in PATH (and LIB/INC/etc.)
- On Windows:
  - **WinSDK** Microsoft C compiler and Windows headers only
  - **Cygwin** Full environment – mingw-w64 cross-compiler, make, bash, etc.
  - **MSYS2** (active) fork of Cygwin – mingw-w64 cross-compiler and more libs
  - **WSL** Unaltered Linux distribution (initially Ubuntu only).
    *Experimental fork available; but presently only works as **true** cross-compiler (i.e. using opam-cross repositories)*

cmd vs PowerShell vs Cygwin vs MSYS2 vs WSL

- cmd – native Windows text shell/terminal (oldest)
- PowerShell – alternative Windows text shell; .NET implementation
- Cygwin – aims to provide a complete *login shell* experience
- MSYS2 – forks Cygwin to provide a complete mingw-w64 compiler set
- WSL – emulates Linux kernel (WSL1) or spins VM (WSL2). “Just a VM”

These choices have been ignored so far

- C compiler

  gcc vs clang / 32-bit

- C runtime library

  glibc vs musl

- OS package manager

  MacPorts vs Homebrew

- Docker…

Windows users more actively need those choices!

Windows needs

- Switch invariants (done for 2.1!)
- Explicit dependencies
- Package parameters
- depext lifting to availability (done for 2.1!)
- layered switches
- Build environments

switch invariants

- opam 2.0

```
opam switch create playground ocaml-base-compiler.4.10.0
```

Installs 6 packages and locks the switch (base-bigarray.base, base-threads.base, base-unix.base, ocaml.4.10.0, ocaml-base-compiler.4.10.0, ocaml-config.1)

- opam 2.1

```
./opam switch create playground --formula='"ocaml" {>= "4.10.0" & < "4.11.0~"}'
```

Requires an ocaml package in the 4.10 series which all upgrades must satisfy

switch invariants – Windows

- Dynamic (shared) linking on Windows is hard
- Since OCaml 3.11, we use a custom linker called flexlink
- It's written in OCaml…
- It can be upgraded separately from OCaml…
- … thanks to switch invariants, the compiler can depend on it but the package can be upgraded in the normal way

Explicit dependencies

- New predicate `{explicit}` allowing, say, the ocaml package dependency to be of the form

```
depends: ocaml-base-compiler | ocaml-system | ocaml-variants {explicit}
```

- The semantics are that the ocaml-variants is only a dependency if it's ***explicitly*** installed – the atom is removed from the formula otherwise.
- The effect is that `opam install ocaml` now cannot select `ocaml-variants` (neither can `opam upgrade`).
- Only an explicit `opam install ocaml-variants.4.10.0+32bit` selects a version.
- Can be implemented with switch invariants, but in the `depends` field, not the switch

Explicit dependencies – Windows

- Multiple C compilers available, which the ocaml-base-compiler will need to depend on (e.g. `conf-msvc | conf-gcc`)
- Need the user's choice to be stable – i.e. the solver shouldn't arbitrarily decide to change the C compiler
- However, the user could choose to: `opam install conf-gcc` should upgrade (😉) OCaml from an msvc port to a mingw port and recompile all packages

Package parameters

- A mechanism is needed to allow packages to receive information from the user at installation
- Prototyped adding --set package:name=value as a parameter both to opam switch create and opam install
- Package commands receive these as environment variables
- Packages can *choose* to persist them in .config files
- package.config files survive opam reinstall
- Many, many, many ocaml-variants packages get nuked
- Now: `opam switch create ocaml-4.10-flambda ocaml.4.10.0 --set ocaml:flambda=true`

Package parameters – Windows

- Used to be a key feature – in 2015 prototypes this was how ports were selected
- Reducing the number of `ocaml-variants` is going to ease upstreaming the Windows build instructions
- Its main use now is to specify the architecture (i.e. i686 or x86_64)
  although this could also be done with a `conf-` package

`depext` lifting from `depends` to `availability`

- Need to be able to refer to depexts through availability, rather than depends, so that the C compiler conf- packages get pulled in only if available (i.e. msvc is either installed or it's not)
- Should be doable with the opam 2.1 feature + extra probes (i.e. Windows-specific parts of interrogating what's installed)
- The key point for Windows is that `opam install conf-msvc ocaml` wants to fail if MSVC is not installed already

Layered switches

- Basic idea is to allow binaries (i.e. programs) from a switch to be available in PATH from other switches
- Proposal is that `opam install --global ocamlformat` puts `bin` files in `gbin`
- opam maintains a hierarchy of switch gbin directories underneath the current switch's bin directory (like `opam remote`)
- For Windows, this means that flexlink can be built in one switch and used by all switches (speeds up switch creation)

Build Environments

- Currently, everything is run in the “system” build environment (modulo the sandbox)
- Idea is to generalise this to allow a build environment to be attached to a switch
- On Windows, would allow, for example, opam to maintain a separate Cygwin installation for doing building, or to switch between Cygwin32 and Cygwin64 if required.
- On Unix (and indeed on Windows), it provides a principled way to have the build for a switch done in a Docker container, with the results exported (or on another build server, etc.)

Where are we now?

- fdopen's fork provides:
  - OCaml installations under the ocaml-variants for i686/x86_64 mingw/msvc with/without flambda pre-compiled/from-source – i.e. 16 packages per release (although he doesn't build msvc flambda for some reason)
  - A Cygwin installation (in C:\OCaml64)
  - A patch on top of depext to drive Cygwin's setup program to install libraries
  - A command line utility (opam-env) in order to drive build systems outside Cygwin

Upstream proposals:

- opam 2.2 would provide:
  - ocaml-base-compiler and ocaml-system support (pre-compiled builds could be officially supported in another remote)
  - A build environment allowing Cygwin/MSYS2 to be in ~\.opam instead
  - depext support for that build environment
  - Native Windows opam, so no need for a shell wrapper. A first time Windows users would not be expected at any point to see a bash terminal

What's critical

- Native shell integration – dra27's final patches
- Switch invariants – **done!**
- Explicit dependencies – *can live without, but…*
- Package parameters – … I'm not sure we'll survive the package explosion
- depext lifting to availability – **done!**
- layered switches – *less critical now for Windows, possibly more urgent elsewhere!*
- Build environments – first-time experience (download opam, run `opam init`)