OCamlAutobuild

OCamlAutobuild
Creating a full build system with a single file

Sylvain Le Gall <sylvain.le-gall@ocamlcore.com>
Presentation at Caml Consortium Meeting
December 4, 2009

OCamlAutobuild: begin...

- Started in July 2008
  - Use pieces of code gathered from 3 other small projects
- September 2008
  - Summary of problems that a Debian packager has to deal when packaging an OCaml library (blog article)
  - Mauricio Fernandez explains Cabal to me and provides a small implementation
- Since then: working on the project from time to time, including features when needed

OCamlAutobuild: Debian problems

After years of Debian OCaml maintainership:

- Missing “static link clause” in LGPL license
- Forgets about non-native architecture
- Missing or incomplete META file
- Doesn't use ocamlfind to install/use libraries
- Doesn't distribute .mli/.cmx files
- Uses bad wildcard in sh/Makefile
- Custom build system
- Missing BTS

Due to the static compilation, it makes LGPL useless.

It makes buildd fail. Requires a fix done by the DD and integrated upstream.

DD create a Debian specific one.This can lead to different names on different distributions (see camlzip).

Libraries get/should be installed in OCaml standard library (not the case in Debian).

Without .mli you are missing a readable header for the library. Without .cmx you are missing some optimizations.

On non native-arch “*.cmxa” and “*.cmx” in shell command lead to errors.

All previous fixes, take longer to find and maintain with a custom build system.

BTS allows to share patches, not only with upstream but with other packagers.
No BTS means less cooperation.

OCamlAutobuild: first draft

We need at least the following steps:

- Configure: checks build environment, allows to disable/enable features
- Build: creates libraries and executables
- Install: moves everything to the right place

We can use

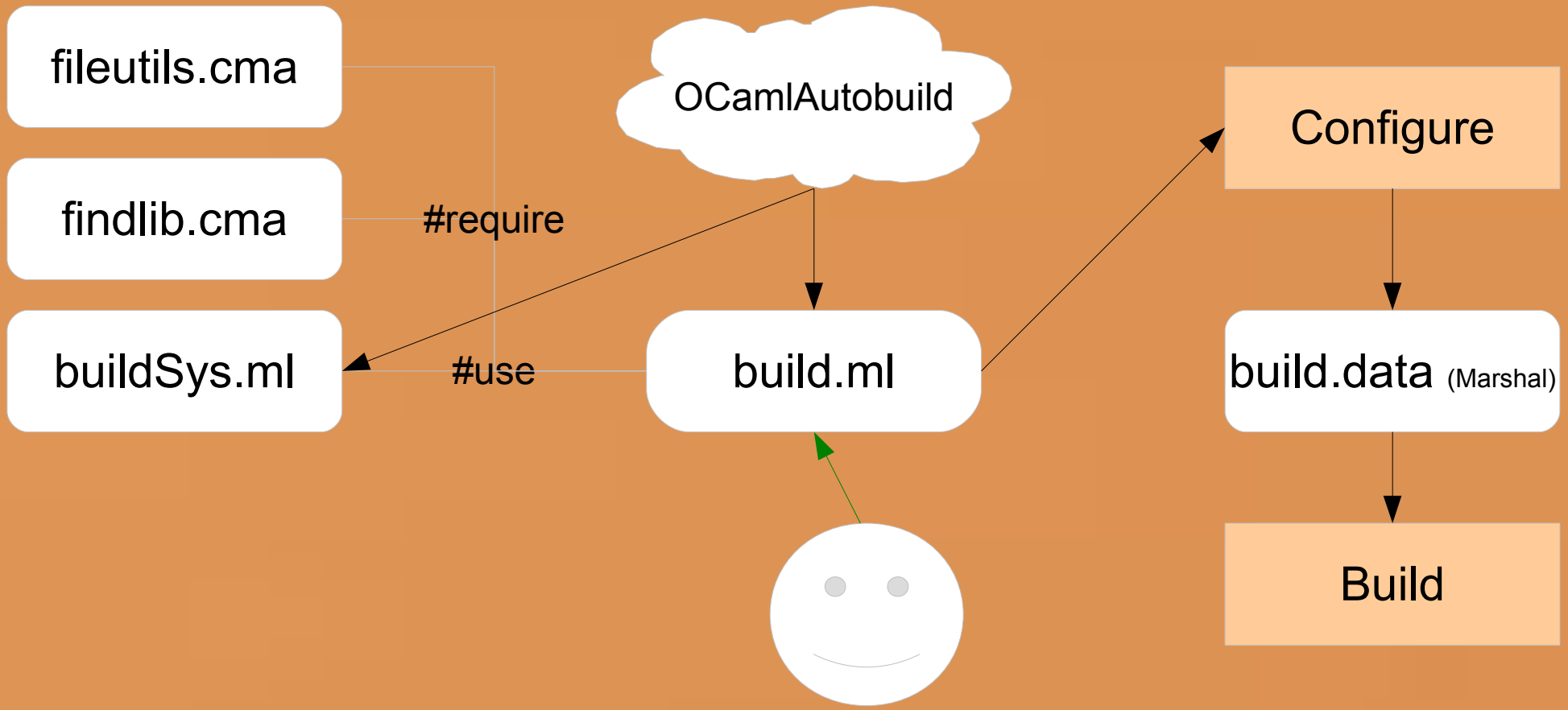
- OCaml as a scripting language
- Findlib to manage libraries
- OCamlbuild, OMake, OCamlMakefile

OCamlAutobuild: first draft

We should avoid:

- Shell scripts and Unix commands
- Adding dependencies
- Forcing projects to change things that works
- Reinventing the wheel

OCamlAutobuild: first draft



OCamlAutobuild: first draft

- Work for small projects
- It doesn't scale when there is a lot of customization to do
  - We fall into the “custom build system”
- Dependencies on findlib.cma and fileutils.cma cannot be fulfilled on Cygwin
- Need to copy and paste a lot of code from one project to another

OcamlAutobuild: Cabal!

Cabal is a system for building and packaging Haskell libraries and programs. It defines a common interface for package authors and distributors to easily build their applications in a portable way

The Cabal also specifies some infrastructure (code) that makes it easy for tool authors to build and distribute conforming packages.

http://www.haskell.org/cabal/

OcamlAutobuild: Cabal!

- This is a building brick of Hackage (CPAN for Haskell)
- It makes really easy to use external libraries
- It is based on a single text file: pkg.cabal
- It simplifies a lot distribution of Haskell software

Reading Cabal user's guide is enlightening:

Simple and Useful

OcamlAutobuild: Cabal!

```
Name:                DecisionTree
Version:             0.0
Cabal-Version:       >= 1.2
Synopsis:            A very simple implementation of decision trees for
                     discrete attributes.
Description:         A very simple implementation of decision trees, built
                     with ID3. You can use it to classify data with a set
                      of discrete attributes.
License:             LGPL
License-file:        LICENSE
Author:              Adrian Neumann
Homepage:            http://page.mi.fu-berlin.de/~aneumann/decisiontree.html
Category:            Algorithms, Pattern Classification
Maintainer:          aneumann@inf.fu-berlin.de
stability:           alpha
build-type:          Simple
extra-source-files:  README, test.hs, lgpl-3.0.txt

Library
    exposed-modules:     Data.DecisionTree
    build-depends:
         base,
         containers >=0.2.0.0
    GHC-Options:    -O2
```

Source: DecisionTree

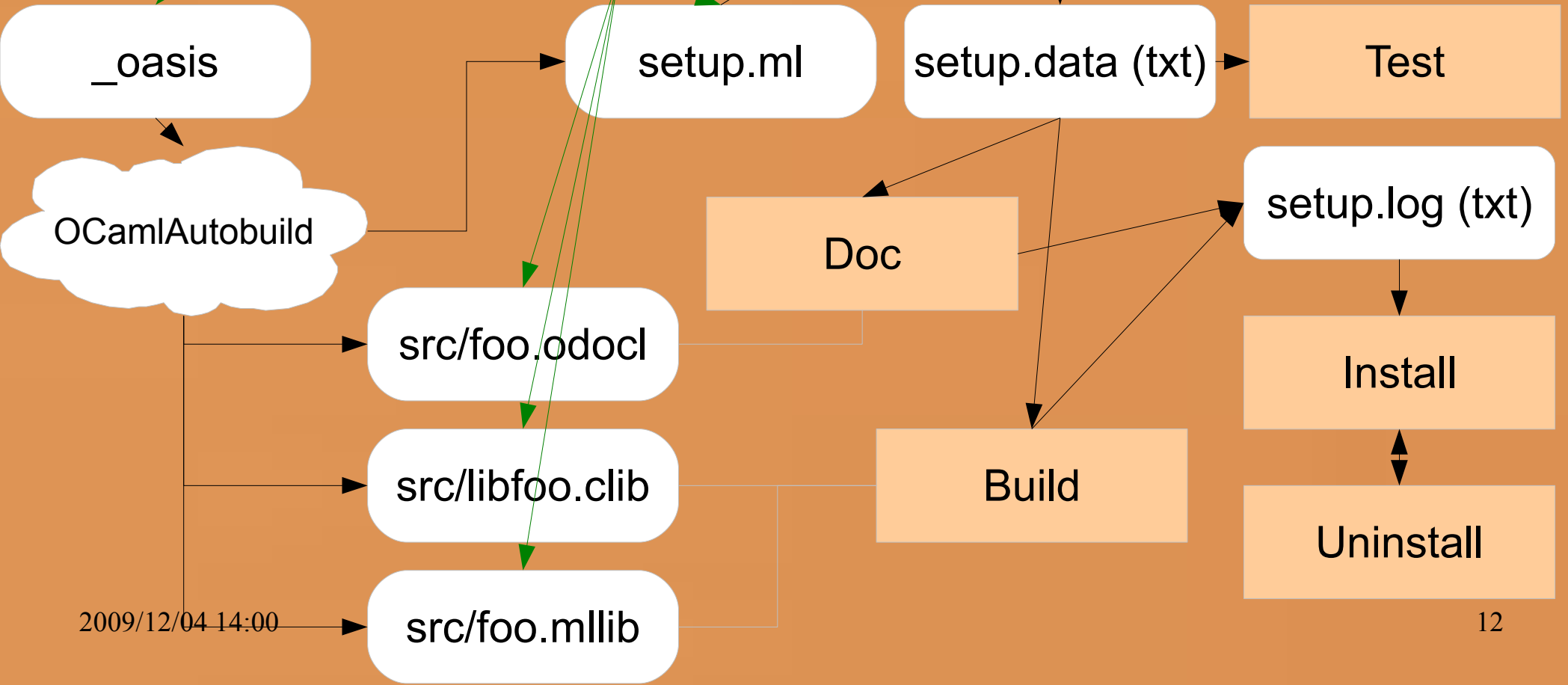
OCamlAutobuild: 2nd version

- "build.ml" is renamed "setup.ml" (c.f. Setup.hs)
- "setup.ml" only depends on OCaml stdlib (no Unix, no FileUtil, no Findlib)
- It uses a simple text file with syntax taken from Cabal: "_oasis".
- It copies everything that is common with Haskell from Cabal user's guide.
- It runs external commands when stdlib is not enough (ocamlfind, 'ocamlc -config', cp, rm...)

OCamlAutobuild: 2nd version

Std user customize *_oasis*

Advanced user customize...



OCamlAutobuild: 2nd version

```
OASISFormat: 1.0
Name:        with-c
Version:     0.0.1
Authors:     Sylvain Le Gall
LicenseFile: LICENSE
License:     LGPL-link-exn
Synopsis:    Minimal project with C file.
Plugins:     META

Library "with-c"
  Path: src
  Modules: A
  CSources: A_stub.c

Executable "test-with-c"
  MainIs: src/main.ml
  CompiledObject: byte
  BuildDepends: with-c
  CSources: main_stub.c

Executable "test-with-c-custom"
  MainIs: src/main_custom.ml
  CompiledObject: byte
  Custom: true
  BuildDepends: with-c
  CSources: main_custom_stub.c
```

Remind you of something ?

OCamlAutobuild: 2nd version

- Scale well from small libraries to projects with several libraries and executables
- Easy to extend through plugins:
  - Translates “_oasis”
  - Embeds code in “setup.ml”
  - Generates files at compile time (“src/foo.odocl”)
- Still needs to create a huge “setup.ml” (~45kB) which is a copy of several small pieces of code.

OCamlAutobuild: 2nd version

- Missing “static link clause” in LGPL license
- ~~Forgets about non-native architecture~~
- ~~Missing or incomplete META file~~
- ~~Doesn't use ocamlfind to install/use libraries~~
- ~~Doesn't distribute .mli/.cmx files~~
- ~~Uses bad wildcard in sh/Makefile~~
- ~~Custom build system~~
- Missing BTS

Plugin META

Plugin internal install

Internal wildcard to install data files

If widely adopted

http://forge.ocamlcore.org

OCamlAutobuild: Projects

- oasis-website: build a website using data contained in “_oasis”
- oasis-selfcontained: create .tar.gz containing everything required to build
- Bocage:
  - Detects upload of source files on OCaml forge
  - Analyzes content to find “_oasis” files
  - Translates “_oasis” to GODIVA
  - Publishes GODIVA file inside GODI

OcamlAutobuild: Conclusion

- Still a lot of work to do (OMake, OcamlMakefile)
- Multi-level marketing:
  - Direct usage for upstream author (build system)
  - Ease packager work
  - Building brick to enhance GODI (through GODIVA)
- Big benefits of reusing Cabal user's guide

OCamlAutobuild

Questions ?

OCamlAutobuild

Demonstration