Développements d'OCaml à l'INRIA

Xavier Leroy

INRIA Rocquencourt

24 janvier 2002

OCaml version 3.04

La version courante, sortie en décembre 2001.

Principales nouveautés:

- Évolution et stabilisation des “labels” sur les paramètres de fonctions.
- Chargement dynamique de la partie C des bibliothèques mixtes Caml/C.
- Intégration du préprocesseur Camlp4.
- License moins restrictive pour les bibliothèques.

Évolution et stabilisation des labels

- Avant: labels optionnels par défaut, beaucoup de labels dans la bibliothèque standard pour documentation.

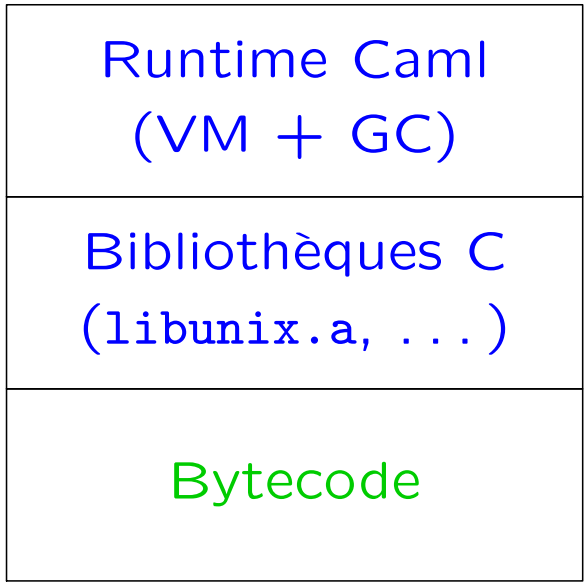
- Maintenant: labels obligatoires par défaut, pas de labels dans la bibliothèque standard, version "labelisée" de certains modules (e.g. `StringLabels` pour `String`).

- Impact: compatibilité totale avec OCaml 2 (sans labels); incompatibilités mineures avec OCaml 3.02 qui affectent les *early adopters* des labels.

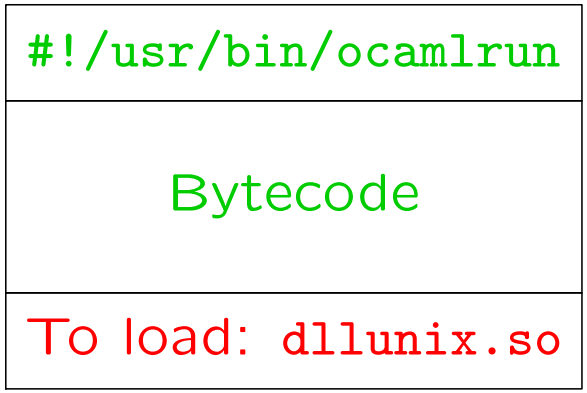
Chargement dynamique de bibliothèques C

Pour les programmes utilisant des bibliothèques mixtes Caml/C (`Unix`, `Str`, `Labltk`, ...).

Avant



Maintenant



L'exécutable bytecode est indépendant de la machine cible.

On peut charger dynamiquement les bibliothèques mixtes (au toplevel ou avec `Dynlink`).

Intégration de Camlp4

Un outil générique pour

- étendre la grammaire du langage
- changer la syntaxe concrète (syntaxe “révisée”, JoCaml)
- imprimer de la syntaxe abstraite.

L’intégration dans OCaml 3.04:

- installation automatique par défaut
- se “branche” mieux dans le toplevel
- les *streams* et *stream parsers* sont gérés par Camlp4.

Changement de la license

License: toujours QPL pour le compilateur + LGPL pour les bibliothèques.

Problème avec la *relinking clause* de la LGPL: oblige le développeur à donner les moyens de re-linker son code avec une autre version de la bibliothèque $\longrightarrow$ doit distribuer sur demande les fichiers objets de son programme.

Maintenant: ajout d'une "exception spéciale" à la LGPL autorisant le linking avec les bibliothèques Caml sans aucune contrainte sur les fichiers de l'application.

Question: est-ce suffisant? faut-il une license spéciale pour les membres du Consortium?

Développements envisagés ou en cours

- Sucre syntaxique: a `` `fn` `` b équivalent à `fn` a b.
- Outil de groupage hiérarchique d'unités de compilation: prend `A.cmo`, `B.cmo`, ... et construit `M.cmo` ayant `A`, `B`, ... comme sous-modules.
- API pour manipuler les grands fichiers ($\geq$ 1Go): représentation des tailles et positions en entiers 64 bits.
- Dé-récursivation du GC mineur.
- Implémentation plus efficace des paresseux (module `Lazy`).
- Nombres complexes dans `Bigarray`.
- Systèmes embarqués: iPaq/Linux, QNX, WinCE?

Travaux de recherche en cours

- Typage polymorphe des arguments des méthodes.

```
class ['a] list = object
  method map: ∀'b.  ('a -> 'b) -> 'b list
end
```

  (Permet de simuler le sous-typage par du polymorphisme de rangée de manière beaucoup plus générale.)

- Réimplémentation de JoCaml (le calcul distribué Join) comme extension d'OCaml (parser avec Camlp4; modifs mineures du typeur).

- Envisagé: vers une intégration de XDuce (transformations et typage fin des données XML) comme extension d'OCaml?