# Ajouter la possibilité de greffons dans un programme en C

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 mars 2007. Dernière mise à jour le 4 septembre 2008

https://www.bortzmeyer.org/greffon-en-c-avec-dlopen.html

---

Aucun programme ne fait tout ce que voudraient ses utilisateurs. C'est pour cela que les programmes à succès ont souvent une possibilité de **greffons**, du code qui est développé entièrement à l'extérieur du programme mais qui peut être exécuté par lui et étendre ses possibilités. Comment faire des greffons en C ?

Un greffon ("*plugin*") est un code qui n'est pas mêlé au source du programme principal et peut donc être développé indépendamment de lui, par un autre auteur, sans nécessiter de coordination ou de système de gestion de sources partagé. En outre, le greffon peut souvent être chargé (et parfois déchargé) dynamiquement. De nombreux programmes réussis ont un système de greffons. C'est le cas de Gimp, Emacs, Apache (qui les appelle « modules »), ...

Dans un langage très dynamique comme l'Emacs Lisp avec lequel est étendu Emacs, charger un greffon est assez facile. Mais comment permettre à un programme C de réaliser cela ? Grâce à `dlopen`. Cette fonction du chargeur permet d'ouvrir un fichier objet et de le charger en mémoire. Par exemple, le programme suivant permet de charger un greffon :

```
#include        <dlfcn.h>
...
int
main (const int argc, const char *argv[])
{

  char *plugin_name;
  char file_name[80];
  void *plugin;
...
  plugin = dlopen (file_name, RTLD_NOW);
  if (!plugin)
    {
      fatal ("Cannot load %s: %s", plugin_name, dlerror ());
    }
```

Le programme utilisant `dlopen`, il doit être lié à la bibliothèque `dl` ainsi :

```
cc  -ldl -o program program.o
```

S'il n'arrive pas à charger le greffon, il produit une erreur. Le greffon est désigné par son nom de fichier et est donc cherché par les mécanismes habituels du chargeur (utilisation de la variable d'environnement `LD_LIBRARY_PATH` sur Unix, par exemple). Voici un exemple de lancement :

```
% ./program -m ./time
Result of plugin ./time is 1173351477
```

Mais, une fois chargé, que fait le programme avec le greffon ? Eh bien, c'est là le point subtil ; le programme doit connaitre les fonctions disponibles chez le greffon. En d'autres mots, il faut définir une API que le greffon va mettre en œuvre et que le programme connaitra. Supposons une API triviale, documentée en tête du programme :

```
  /* The functions we will find in the plugin */
  typedef void (*init_f) ();
  init_f init;
  typedef int (*query_f) ();
  query_f query;
```

Cette API dispose de deux fonctions, `init` qui initialise le greffon (c'est un pointeur sur une fonction puisque les fonctions ne sont pas des objets de première classe en C), et `query` qui fait le vrai travail et qui renvoie un entier (`int`). Les paramètres pris en entrée par le greffon ne sont pas indiqués (nous sommes en C, après tout).

Armé de cette API, notre programme peut l'utiliser :

```
  init = dlsym (plugin, "init");
  result = dlerror ();
  if (result)
    {
      fatal ("Cannot find init in %s: %s", plugin_name, result);
    }
  init ();
```

`dlsym` lui permet de récupérer l'adresse de la fonction en connaissant son nom. Ensuite, il n'y a plus qu'à l'appeler.

L'autre fonction est plus intéressante puisqu'elle renvoie une valeur :

```
  query = dlsym (plugin, "query");
  result = dlerror ();
  if (result)
    {
      fatal ("Cannot find query in %s: %s", plugin_name, result);
    }
  printf ("Result of plugin %s is %d\n", plugin_name, query ());
```

Comme on le voit, ce n'est pas plus difficile.

Le programme complet, utilisant la bibliothèque `popt` pour gérer les arguments, est disponible en ligne (en ligne sur `https://www.bortzmeyer.org/files/dlopen.tar.gz`). On peut aussi trouver un programme réel (mais encore relativement simple) qui utilise `dlopen` : echoping <`http://echoping.sourceforge.net/`>, depuis sa version 6.

La conception de l'API des greffons est le plus gros travail. Comme en général, les greffons vont être développés par d'autres que l'auteur du programme, il faut soigner la documentation.

Et dans d'autres langages ? Un mot de Python. Étant un langage dynamique, il est facile de charger un greffon en évaluant une chaîne de caractères qui est une instruction Python :

```
    try:
        exec ("from " + plugin_name + " import Plugin")
        myplugin = Plugin(module_options, config=config)
    except ImportError, message:
        fatal ("No such module " + plugin_name + \
               " (or no Plugin constructor) in my Python path: " + str(message))
    except Exception:
        fatal ("Module " + plugin_name + " cannot be loaded: " + \
               str(sys.exc_type) + ": " + str(sys.exc_value) + \
               ".\n    May be a missing or erroneous option?")
```

Le greffon doit hériter d'une classe qui définit l'API nécessaire (ou fournir directement l'API demandée par le programme ci-dessus). Un exemple d'un tel greffon, lorsque le programme demande une méthode `query()` est :

```
class Plugin:

    def __init__ (self):
        pass

    def query(self, arg):
         ...
```

Le programme pourra alors appeler le greffon qu'il a chargé plus haut :

```
    result = myplugin.query("something")
```

En Haskell, hs-plugins <`http://www.cse.unsw.edu.au/~dons/hs-plugins/`> est un mécanisme de développement de greffons. "*Dynamic Applications From the Ground Up*" <`http://www.cse.unsw.edu.au/~chak/papers/yi.ps.gz`> est un excellent article décrivant une architecture de greffons pour Haskell.

Enfin, pour l'anecdote, on notera que l'Office de la Langue Française au Québec prétend traduire "*plugin*" par le ridicule plugiciel <`http://www.olf.gouv.qc.ca/RESSOURCES/bibliotheque/dictionnaires/Internet/fiches/1299146.html`>, pourtant un anglicisme alors que **greffon** est bien établi depuis longtemps.

Sinon, cet article décrit une méthode où le programmeur fait tout, directement avec `dlopen` et `dlsym`. Mais il peut aussi utiliser une bibliothèque qui l'aide comme C-Pluff <`http://www.c-pluff.org/`> (que je n'ai pas testé).

---

`https://www.bortzmeyer.org/greffon-en-c-avec-dlopen.html`