Créer un plugin XMMS

par Romain Guy (Gfx)

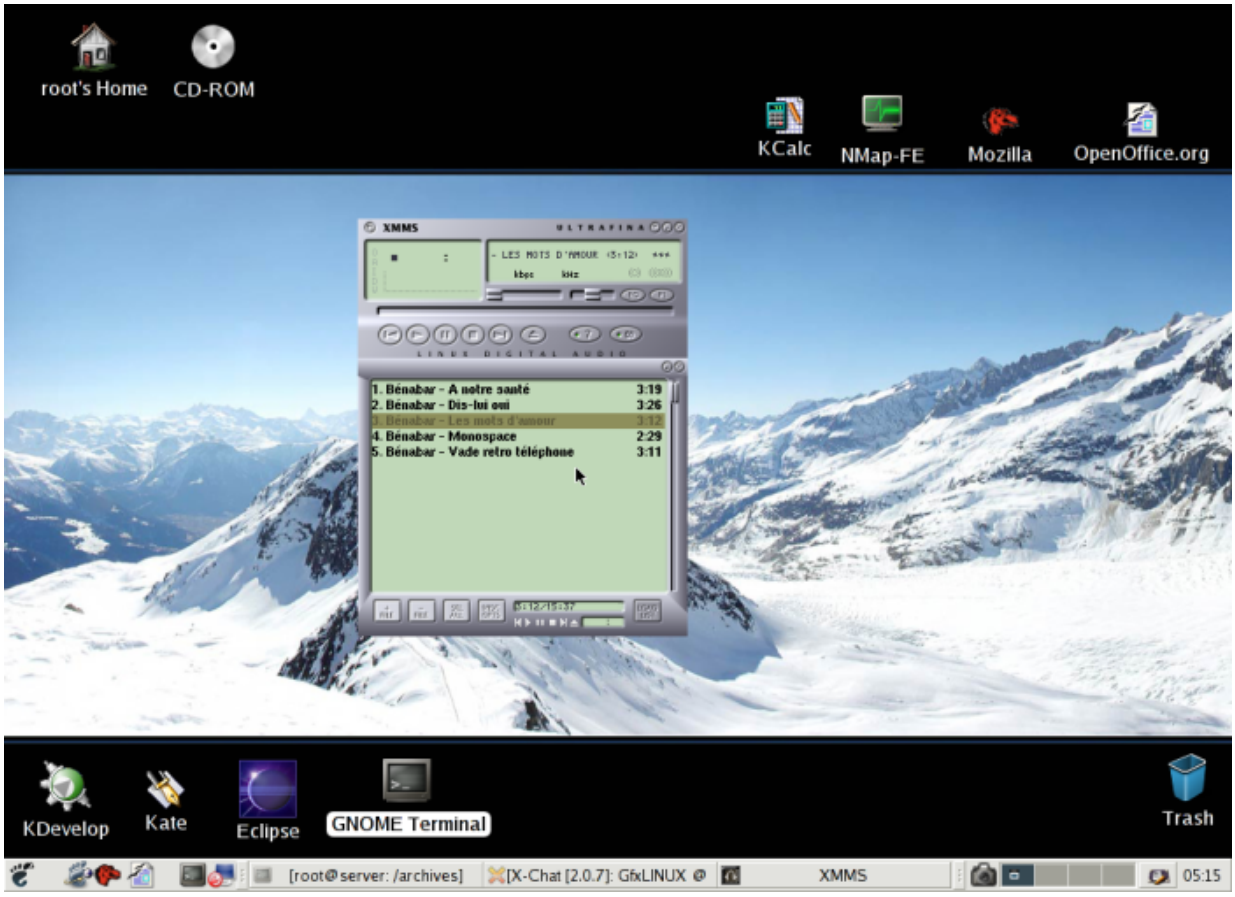
Date de publication : 6 Juin 2006

Dernière mise à jour :

XMMS est un lecteur multimédia sous licence GPL pour système Linux. Ce programme agréable et puissant peut se voir amélioré par l'entremise de plugins. Découvrons ensemble comment en créer un.

I - Introduction

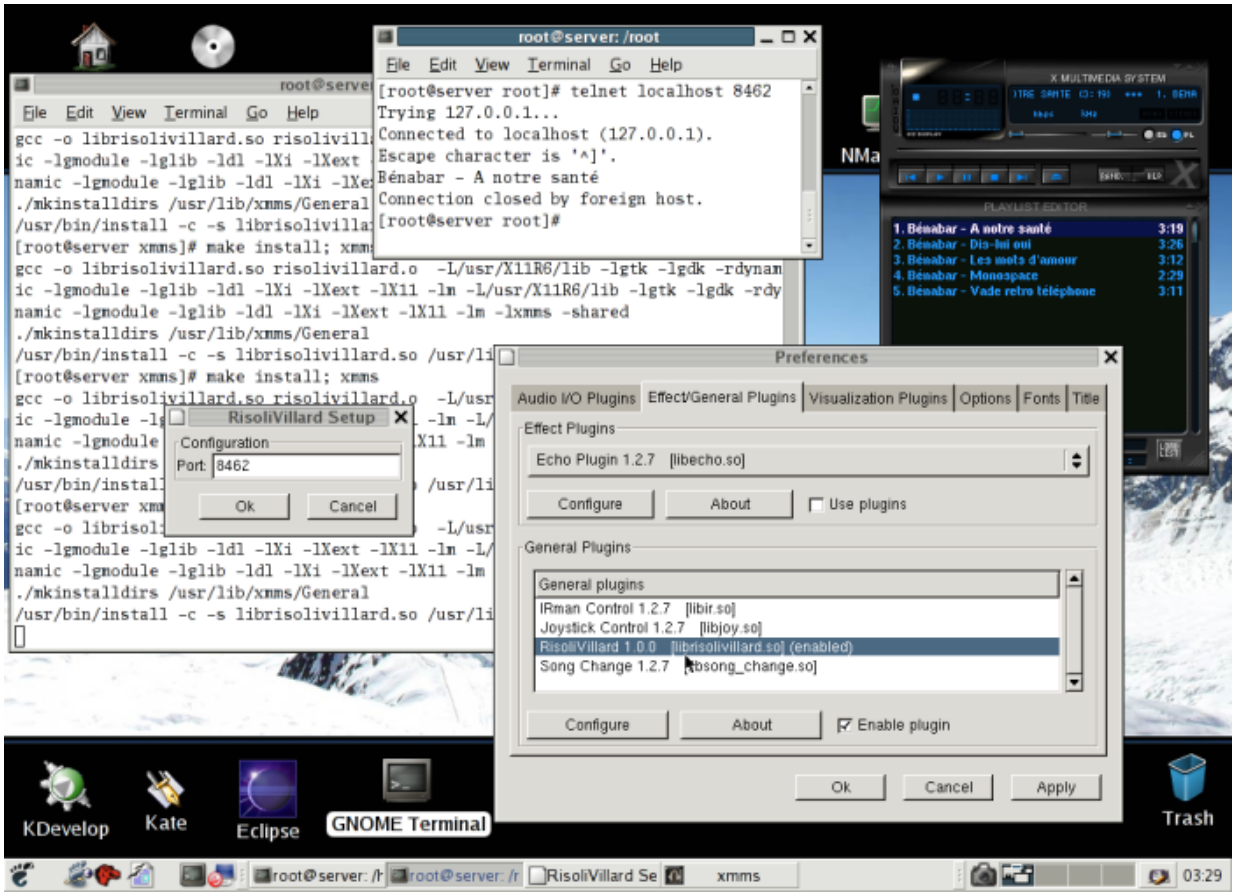
Les plugins constituent aujourd'hui l'une des caractéristiques les plus importantes des lecteurs de musique numérique. Très souvent exploités par les auteurs de ces logiciels pour fournir les fonctionnalités de base, il sont également très appréciés des autres développeurs qui peuvent ainsi personnaliser pleinement leur outil favori. Nous allons apprendre à réaliser notre propre extension pour XMMS à travers la distribution RisoliVillard. Cette dernière contient un ensemble de plugins et de scripts pour différents logiciels et différents systèmes d'exploitation. Son principe est simple : un lecteur de musique équipé d'une extension RisoliVillard ouvre un port en local (par défaut 8462) sur lequel tout le monde peut se connecter. Dès la connexion, le titre de la chanson actuellement jouée est envoyé au client et la connexion est fermée. De nombreux usages sont alors possibles allant de l'affichage sur votre site Internet personnel à la diffusion de vos chansons sur IRC en passant par l'insertion dans la signature de vos courriers électroniques. Nous testerons d'ailleurs notre nouveau plugin en écrivant un petit script pour le client IRC XChat.



XMMS figure parmi les lecteurs multimédia les plus populaires sous Linux.

II - Anatomie d'un plugin

La première étape du développement consiste à installer la libxmms et ses en-têtes sur votre machine que vous les trouverez dans la plupart des distributions Linux. Nous devons ensuite choisir la nature de notre extension. En lisant attentivement le fichier /usr/include/xmms/plugin.h vous découvrirez que nous pouvons créer des OutputPlugin, des EffectPlugin, des InputPlugin, des GeneralPlugin et des VisPlugin. Etant donné la raison de notre travail nous devons choisir de créer un GeneralPlugin. Le listing 1 présente la structure correspondante. En compilant notre plugin sous la forme d'une bibliothèque partagée /usr/lib/xmms/General/librisolivillard.so nous allons demander à XMMS d'exécuter notre fonction get_gplugin_info() au démarrage. Cette dernière doit retourner un pointeur sur la structure décrivant notre plugin, ainsi qu'en témoigne le listing 2. Les trois premiers membres de la structure sont manipulés directement par XMMS et ne doivent pas être renseignés. Notez la présence de xmms_session qui identifie votre plugin auprès des différentes fonctions de contrôle. Les quatre pointeurs sur fonction correspondent pour leur part aux différentes possibilités qu'offre votre extension. Les pointeurs init et cleanup se révèlent indispensables puisqu'ils contrôlent l'initialisation et la fin du cycle de vie de votre plugin. Si vous attribuez NULL aux pointeurs about et configure, les boutons correspondants dans les options de XMMS apparaîtront grisés. Nous vous conseillons de les ignorer pour le moment. Une fois votre plugin compilé et copié dans le répertoire /usr/lib/xmms/General, vous pouvez lancer XMMS pour vérifier qu'il se charge correctement en vérifiant qu'il apparaît dans l'onglet "Effect/General Plugins" des options.



Notre plugin peut être configuré directement depuis XMMS

L'étape suivante consiste à mettre en place un serveur sur le port 8462 qui servira chaque client se connectant sur ce port. Nous allons donc mettre en place un thread qui sera chargé d'attendre indéfiniment des requêtes clientes. Ce dernier sera mis en place par la fonction d'initialisation risolivillard_init() et exécuté dans risolivillard_thread(). Puisque la réponse que nous devons fournir à chaque client n'est autre que le nom de la chanson actuellement

jouée par le lecteur, nous devons la récupérer à l'aide des fonctions de contrôle de XMMS. Celles-ci se trouvent dans l'en-tête xmmsctrl.h et permettent de manipuler entièrement le logiciel. Nous pouvons ainsi basculer le mode de lecture aléatoire grâce à la fonction xmms_remote_toggle_shuffle() ou encore modifier le volume avec xmms_remote_set_main_volume(). En étudiant l'ensemble des fonctions proposées, vous constaterez qu'il n'existe aucun moyen pour accéder directement au nom de la chanson écoutée. XMMS fonctionne en effet sur le principe de playlist et la chanson lue y figure nécessairement. Nous devons donc simplement connaître la position de celle-ci puis récupérer le titre de l'élément de la playlist correspondant à cette position en invoquant les fonctions xmms_remote_get_playlist_pos() et xmms_remote_get_playlist_title(). Le listing 3 démontre comment parvenir à envoyer le titre au client. Nous retrouvons ici l'attribut xmms_session que nous avions évoqué précédemment : vous en aurez besoin pour chaque appel aux fonctions xmms_remote_*. Nous pouvons aisément valider notre code en exécutant telnet localhost 8462 depuis la console et vérifier que le nom de la chanson s'affiche.

Si notre extension fonctionne à merveille, un gros défaut subsiste. Il est en effet probable que les utilisateurs souhaitent pouvoir configurer le numéro de port utilisé afin de passer outre un pare-feu ou un proxy récalcitrant. XMMS propose à cet effet une API de lecture et d'écriture de préférences dans des fichiers, accessible en incluant l'en-tête configfile.h. Les fichiers lus et générés de la sorte sont de simples fichiers de type "clé=valeur". Le listing 4 présente un exemple de lecture dans un tel fichier. Les fichiers peuvent être manipulés n'importe où sur le disque dur mais il est conseillé de les placer dans le répertoire de configuration ~/.xmms. Après ouverture du fichier, nous lisons ici simplement une chaîne de caractère désignée par la clé "rvPort" dans la section "risolivillard". L'API propose également des fonctions pour lire des booléens, des entiers et des nombres décimaux. L'opération d'écriture ressemble beaucoup à cet extrait de code à une subtilité près. Quand l'ouverture du fichier échoue, XMMS ne crée pas automatiquement les ressources nécessaires pour enregistrer les clés et leurs valeurs, vous devrez appeler explicitement la fonction xmms_cfg_new() :

```
if ((config = xmms_cfg_open_file(configfile)) == NULL)
  config = xmms_cfg_new();
// #
xmms_cfg_write_file(config, configfile);
xmms_cfg_free(config);
```

La sauvegarde effective des données n'a lieu pour sa part que lors de l'invocation de xmms_cfg_write_file(). Le code source complet disponible sur le CD-Rom accompagnant ce magazine utilise cette API à travers une interface simple en GTK laissant à l'utilisateur la possibilité de saisir le port à utiliser.

III - Un plugin en Python ?

Si la programmation C vous donne des boutons sachez qu'il existe une autre solution pour parvenir à vos fins. Florent Rougon a développé un module Python intitulé PyXMMS qui permet d'invoquer les fonctions décrites dans xmmsctrl.h depuis un simple script. Les possibilités se révèlent certes plus limitées mais vous pourrez ainsi programmer rapidement de petites applications très utiles. Le listing 5 contient ainsi le code complet d'un script Python qui propose exactement les mêmes fonctionnalités que le plugin XMMS que nous venons d'étudier. En important le module xmms dans votre code, vous aurez accès aux mêmes fonctions que précédemment. Ces dernières perdent au passage leur encombrant préfixe xmms_remote_ ainsi que leur premier paramètre, le fameux xmms_session. Tout pointeur utilisé par ces fonctions pour retourner des informations disparaît également pour devenir une simple valeur de retour dans un tuple. Le script et PyXMMS se trouvent également sur le CD-Rom.

Puisque nous pouvons dorénavant publier le nom des chansons que nous écoutons, pourquoi ne pas en faire profiter tout le monde en les diffusant sur les salons de discussion IRC ? Le client XChat autorisant la réalisation de petites extensions en Python, nous allons en profiter pour découvrir comment utiliser l'extension XMMS que nous venons de créer. Après avoir créé votre script dans lequel vous avez importé le module xchat, saisissez la ligne suivante :

```
xchat.hook_command("music", onNoticeRisoliVillard, help="/music Afficher la chanson")
```

La fonction hook_command() permet de créer une nouvelle action de la forme /commande dont la frappe déclenche l'exécution du gestionnaire associé, ici onNoticeRisoliVillard(), décrit dans le listing 6. Les différents paramètres, inutilisés dans notre cas, correspondent au nombre de paramètres saisis par l'utilisateur, au tableau contenant ces paramètres et à une donnée que vous pouvez vous-mêmes affecter lors de l'exécution de hook_command(). Lorsque notre gestionnaire se déclenche, nous ouvrons un socket en local sur le port 8462 pour y lire immédiatement des données que nous affichons sur le salon courant. Nous utilisons pour ce faire la fonction xchat.command("me message") qui a le même effet que la saisie de /me message par l'utilisateur. Il s'avère donc possible d'utiliser RisoliVillard depuis un script en exécutant xchat.command("music").

```
typedef struct
{
  void *handle;
  char *filename;
  int xmms_session;
  char *description;
  void (*init) (void);
  void (*about) (void);
  void (*configure) (void);
  void (*cleanup) (void);
} GeneralPlugin;
```

```
#include <plugin.h>
#define VERSION "1.0.0"

GeneralPlugin risolivillard =
{
  NULL, NULL, -1, "RisoliVillard "VERSION,
  risolivillard_init, risolivillard_about,
  risolivillard_config, risolivillard_cleanup,
};

GeneralPlugin *get_gplugin_info(void)
{
  return &risolivillard;
}
```

```
for (;;)
{
  // server est le socket ouvert en écoute sur le port 8462
  client = accept(server, NULL, NULL);
  if (client == -1)
```

```
      continue;

    pos = xmms_remote_get_playlist_pos(risolivillard.xmms_session);
    song = xmms_remote_get_playlist_title(risolivillard.xmms_session, pos);
    song = g_strconcat(song, (gchar *) "\r\n", NULL);
    send(client, song, strlen((char *) song), 0);

    close(client);
    sleep(1);
  }
```

```
  void read_config(void)
  {
    gchar *configfile;
    ConfigFile *config;

    configfile = g_strconcat (g_get_home_dir(), "/.xmms/risolivillard", NULL);
    if ((config = xmms_cfg_open_file(configfile)) != NULL)
      xmms_cfg_read_string(config, "risolivillard", "rvPort", &rvPort);
    g_free(configfile);
  }
```

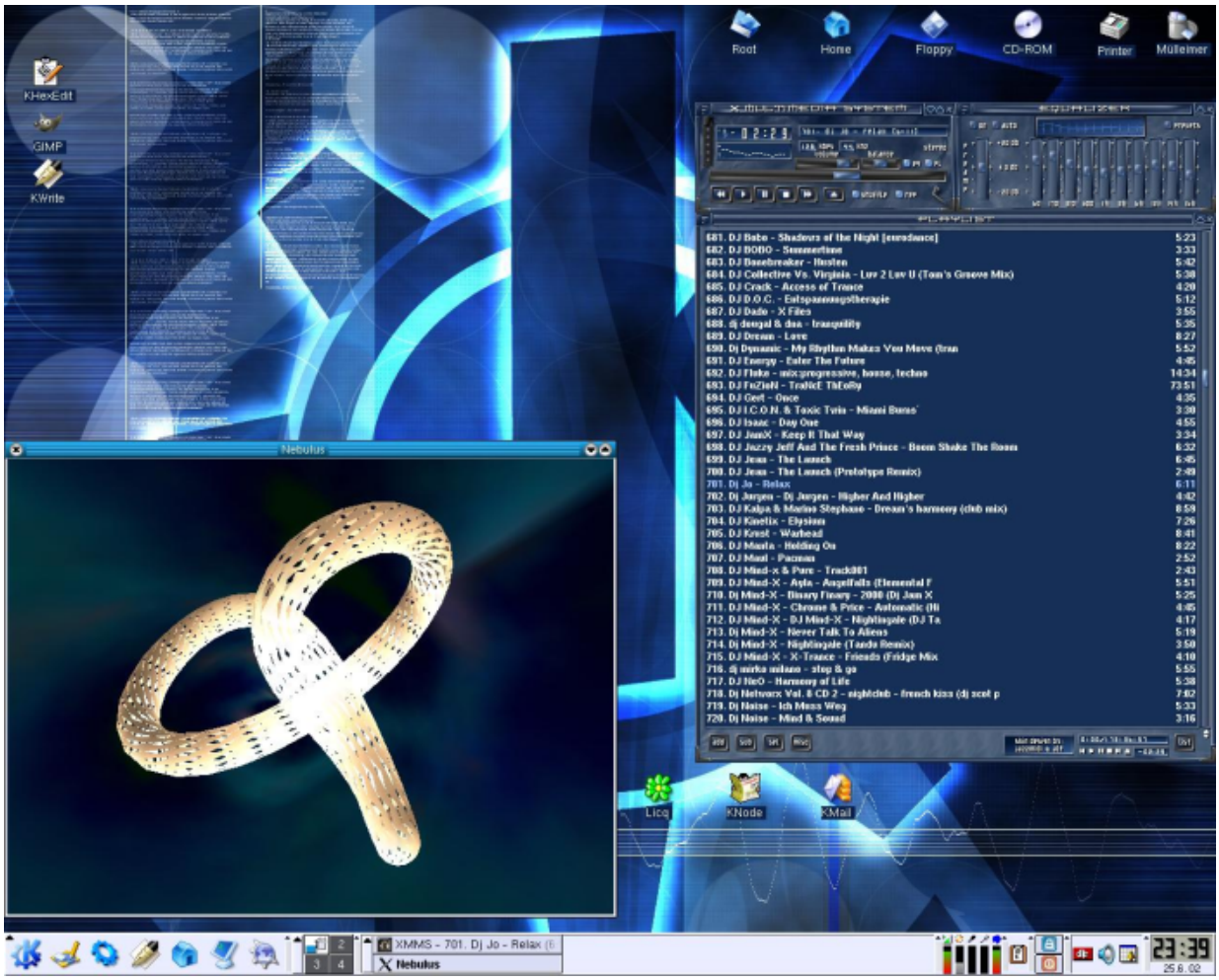
```
  from socket import *
  import xmms

  rv = socket(AF_INET, SOCK_STREAM)
  rv.bind(('', 8462))

  while true:
    rv.listen(1)
    guest, guestHost = rv.accept()
    song = xmms.get_playlist_title(xmms.get_playlist_pos())
    guest.send(song is None and " " or song)
    guest.close()

  rv.close()

  (((listing 6)))
  def onNoticeRisoliVillard(word, word_eol, userdata):
    rv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    rv.connect(('localhost', RISOLIVILLARD_PORT))
    data = rv.recv(1024)
    xchat.command("me %s %s" % (RISOLIVILLARD_MESSAGE, data))
    rv.close()
    return xchat.EAT_NONE
```



Il est bien évidemment possible de réaliser des plugins complexes affichant par exemple des animations 3D.