

# Création et utilisation de DLL dans mIRC

par PINO Bastien

Date de publication : 08/08/07

Dernière mise à jour :

Cet article va vous permettre de réaliser une DLL qui sera utilisable dans mIRC. Vous trouverez le code d'un DLL en PureBasic et en C/C++. Ensuite, vous apprendrez comment faire pour utiliser les fonctions de la DLL dans mIRC. Cette solution va permettre d'étendre les possibilités de mIRC.

- I - Introduction
- II - Utilisation de DLL dans mIRC
- III - Création de la DLL
  - III-A - Généralités et bases
    - III-A-1 - Prototype
    - III-A-2 - Valeurs de retour
  - III-B - Implémentation PureBasic
  - III-C - Implémentation C++
    - III-C-1 - Fichier CPP
    - III-C-2 - Fichier DEF
- IV - Conclusion
- V - Remerciements

## I - Introduction

Comme beaucoup d'autres langages intégrés, le mIRC scripting ne permet pas de faire tout ce que l'on désire. En effet le mIRC script permet de faire beaucoup d'actions sur la gestion des évènements IRC mais dès que l'on veut faire quelque chose d'un peu compliqué on est vite freiné.

Pour palier à ces petits inconvénients, mIRC dispose de différentes solutions, à savoir le DDE (communication interprocessus) et le couplage avec une DLL. Nous allons ici parler des DLLs qui permettent d'ouvrir un monde infini sur les possibilités d'options de vos scripts.

### Qu'est-ce qu'une DLL (*Dynamic Link Library*) ?

Microsoft définit une DLL comme un module qui contient des données et des fonctions. Une DLL est chargée en mémoire par un programme appellant (.exe) ou bien par une autre bibliothèque (.dll). En ce sens, il y a deux types de fonctions dans une librairie, les *internal* et *external*.

- **external** : ces fonctions peuvent être appelées par n'importe quel module.
- **internal** : ces fonctions ne sont accessibles que par le module appellant.

## II - Utilisation de DLL dans mIRC

Avant de développer une DLL, il serait quand même judicieux de savoir comment faire pour utiliser celle-ci. Il n'y a besoin d'aucunes manipulations pour charger une DLL dans mIRC, le simple fait de l'appeler par la fonction du script la charge automatiquement.

Il y a deux possibilités d'appeler une DLL :

### Par appel de commande basique

```
/dll <nomDeLaDLL> <fonction> <paramètres>
```

### Par fonction de scripting

```
$dll(<nomDeLaDLL>, <fonction>, <paramètres>)
```

**Attention :** il y a une différence fondamentale entre ces deux appels. Le premier ne permet pas de récupérer la valeur de retour, tandis que l'autre oui.

### Comment récupérer la (les) valeur(s) retournée(s)

```
var %result = $dll(maDLL.dll,maJolieFonctionQuiRetourne,parametres)
```

Ensuite concernant la mise en mémoire de la DLL, soit elle reste en mémoire soit elle attend que mIRC la décharge. Au bout de 10 min d'inactivité, mIRC essaye de décharger la DLL de la mémoire. Encore une fois, c'est la DLL qui va décider de ce qu'elle veut faire.

Voici un bout de code très connu qui permet de lister toutes les DLL chargées.

### Tappez

```
//var %c = $dll(0) | while %c { echo -a $dll(%c) | dec %c }
```

Si vous voulez forcer une DLL à être déchargée, utilisez l'option -u. Pour décharger toutes les DLL :

### Tappez

```
//var %c = $dll(0) | while %c { dll -u $dll(%c) | dec %c }
```

En ce qui concerne les conventions de nommage, il n'en existe pas vraiment ... Tout cela est laissé à l'appréciation du développeur. [ Petite parenthèse : je profite de ce moment pour vous conseiller de bien réfléchir lors de la conception de votre DLL pour les noms des fonctions de celle-ci ]

## III - Création de la DLL

### III-A - Généralités et bases

#### III-A-1 - Prototype

Voici le pseudo code de l'entête à utiliser pour faire communiquer la DLL avec mIRC :

```
fonction(HWND mWnd, HWND aWnd, char *data, char *params, BOOL print, BOOL nopause)
```

- **fonction** est le nom de la fonction que nous allons utiliser sous mIRC (*ATTENTION à la casse !*)
- **HWND mWnd** est le "handle" de la fenêtre principale de mIRC (c'est l'identifiant de la fenêtre mIRC)
- **HWND aWnd** est le "handle" de la fenêtre active (en général NULL car utilisé par un script)
- **char \*data** contient le pointeur vers les données que l'on envoie à la DLL ( 900 octets max )
- **char \*params** contient les paramètres pour exécuter une commande mIRC ( 900 octets max )
- **BOOL print** contient FALSE si appelé par /.dll pour cacher la commande, sinon contient TRUE
- **BOOL nopause** si ce paramètre contient TRUE, alors mIRC exécute une routine critique.

#### III-A-2 - Valeurs de retour

Les valeurs de retour de *fonction* sont limitées et chacune influent sur le comportement de mIRC.

- **0** Arrête l'exécution du script (/halt)
- **1** mIRC peut continuer à traiter le script
- **2** data est rempli avec une commande et params avec les paramètres
- **3** data contient ce que \$dll() doit afficher

### III-B - Implémentation PureBasic

Créez un projet de DLL, et insérez le code suivant :

#### Squelette d'une procédure de création de DLL pour mIRC

```
ProcedureDLL HelloWorld(mWnd, aWnd, *StringData, *StringParam, show.b, nopause.b)
;ici on récupère les paramètres envoyer via $dll()
host$ = PeekS(*StringData)

;ici on place la chaine dans le tube de sortie
PokeS(*StringData, "HelloWorld ! Arguments DLL : " + host$)

; on retourne 3 car il a un retour dans *StringData
ProcedureReturn 3
EndProcedure
```

On compile ce code sous hello.dll, et l'on place sa DLL dans le répertoire courant de **mirc.exe**. En écrivant l'alias suivant :

#### A mettre dans Alias

```
/testDLL {  
    set %param coucou!  
    echo -a $dll(hello.dll, HelloWorld, %param)  
}
```

On devrait voir apparaître sur la fenetre active :

*HelloWorld ! Arguments DLL :coucou!*

### III-C - Implémentation C++

Créer un projet de DLL et ajouter un fichier DEF à votre solution afin de pouvoir exporter les différentes fonctions de votre composition.

Cet exemple a été fait avec VS, je mettrai à votre disposition plus tard un autre exemple pour DevC++.

#### III-C-1 - Fichier CPP

Dans ce fichier nous allons mettre nos différentes fonctions de notre DLL.

#### Squelette d'une procédure de création de DLL pour mIRC

```
int __stdcall WINAPI HelloWorld(HWND mWnd, HWND aWnd, char *data, char *parms, BOOL print, BOOL  
nopause)  
{  
    char *args;  
    args = malloc(sizeof(char) * strlen(data) + 1);  
  
    strcpy(args, data); // copie des arguments dans une variable tampon  
    strcpy(data, strcat("HelloWorld ! Arguments : ", args)); // copie dans le tube de sortie  
  
    return 3; // on annonce à mIRC qu'il y a des infos à retourner  
}
```

#### III-C-2 - Fichier DEF

**Attention**, votre fichier DEF doit comporter le même nom que celui de la DLL.

Ce fichier DEF va permettre d'exporter l'ensemble des fonctions de votre librairie. Vous devez mettre chaque fonction que vous mettrez dans votre DLL dans ce fichier, sous la forme :

```
LIBRARY NomDeLaDLL  
EXPORTS  
fonction1  
fonction2  
...
```

Dans notre exemple, il faudrait donc rajouter :

```
LIBRARY hello  
EXPORTS  
HelloWorld
```

## IV - Conclusion

Nous voici arrivé à la fin de notre article. Vous avez donc compris qu'il est très simple de fabriquer une DLL pour optimiser les performances et faire des choses que ne permet pas le mIRC script.

Un des exemples classiques est la gestion de la 3D pour les intros, ou encore les possibilités graphiques (docking de popups, ...).

J'espère que cet article vous aura intéressé, n'hésitez pas à me faire des commentaires sur le forum ou à poser des questions sur vos développements.

## V - Remerciements

J'adresse ici tous mes remerciements à l'équipe de rédaction de "developpez.com" pour le temps qu'ils ont bien voulu passer à la correction et à l'amélioration de cet article.

Plus particulièrement à mavina et Swoög pour leur aide et Etanne pour la relecture.

