Jean-Yves Tigli, Stéphane Lavirotte
Polytech'Nice – Sophia
SI3
2017-2018

TD n° 2 bis
Bibliothèques Statiques et Dynamiques sous Win32

1 Environnement de développement et Projets sous Win32

1.1 Installation de Visual Studio

En tant qu'étudiant à Polytech au département Sciences Informatiques, vous avez la possibilité d'utiliser les outils de développement Microsoft gratuitement. Pour cela, il vous suffit de vous connecter sur le site Microsoft Imagine et de vous identifier. Vous pourrez alors télécharger différentes versions de système d'exploitation ou bien les outils de production de Microsoft en toute légalité. Attention c'est un accès personnel et vous ne devez pas fournir ces accès où le code produit que vous obtenez à un tiers. Votre nombre de téléchargement et d'accès aux logiciels est limité. C'est donc pour votre usage personnel pour les cours et TD.

https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=dc5487f7-669b-e011-969d-0030487d8897&vsro=8

Si vous rencontrez un problème, vous pouvez aussi utiliser Visual Studio Community 2015. Cette version de l'IDE est téléchargeable et utilisable gratuitement pour tous.

https://www.visualstudio.com/fr/vs/community/

1.2 Création d'un projet Windows Visual C/C++

Vous allez commencer par créer un Nouveau Projet « *Fichier* / *Nouveau* / *Projet*… ». Vous veillerez à sélectionner l'environnement « *Visual C++* / *Général* » sélectionnerez « *Projet Vide* » en lui donnant le nom « *Bibliotheques* » (évitez d'utiliser des lettres accentuées). Vous aurez alors la création d'une Solution et d'un projet à l'intérieur de cette solution s'appelant Bibliotheques.

Cet IDE n'ouvre qu'une solution à la fois mais qui peut contenir plusieurs projets. Il est donc fortement conseillé de ne pas créer une solution par exercice mais de rajouter un nouveau projet à une solution commune pour tout le TD. Pour ajouter un projet à une solution existante, il suffit de faire un clic droit sur le nom de la solution au niveau de l'« *Explorateur de solutions* » et sélectionner « *Ajouter* / *Nouveau Projet* ». Si vous réalisez cette opération par le menu « *Fichier* / *Nouveau* / *Projet* », par défaut il créé une nouvelle solution (ce qui n'est pas souhaitable pour ce TD). Vous pouvez sélectionner l'option « *Ajouter à la Solution* » dans la partie basse de la fenêtre.

Maintenant que vous savez tout sur la création de Solution et Projet sous Visual Studio, nous allons présenter quelques autres fonctionnalités utiles pour ce TD.

1.3 Utilisation de Visual Studio

La compilation du ou des projets d'une solution (ou de toute la solution) est réalisée grâce au menu « Générer ». Il permet de (re)générer la solution ou le projet et de les nettoyer (efface le contenu du répertoire Debug qui contient les productions `.lib`, `.dll`, `.exe` et autres fichiers spécifiques).

Quand vous avez plusieurs projets exécutables dans une solution, n'oubliez pas de sélectionner le projet à exécuter : « *Projet* / *Définir comme projet de démarrage* » (ou en faisant un clic droit sur le projet en question dans l'explorateur de solutions.

Les dépendances entre projets sont gérées par le menu « *Projet* / *Dépendances du projet* » (il faut bien entendu qu'il existe plusieurs projets dans votre solution). Vous pouvez aussi exprimer cette notion en faisant un clic sur les « Références » d'un projet et demander d'ajouter une référence et ainsi faire référence à un autre projet de la solution ou bien même à un fichier extérieur à votre solution. La création d'une dépendance entre projet va modifier l'ordre de génération des projets. Vous pouvez vérifier cet ordre dans « Projet / Ordre de génération du projet ». Comme cela était le cas avec notre `Makefile` sous Unix, il faut tout d'abord compiler la bibliothèque avant de compiler le programme qui l'utilise.

TD n° 2 bis
Bibliothèques Statiques et Dynamiques sous Win32

Pensez à utiliser l'explorateur de solution dans la fenêtre de droite. Cela affiche l'ensemble des projets de la solution, et permet d'afficher leurs propriétés.

2 Bibliothèque et édition de liens statiques et dynamiques

2.1 Création de la bibliothèque statique

Exercice n°1:

Nous allons configurer le projet « *Bibliotheques* » que vous avez créé. Nous allons faire une bibliothèque de type bibliothèque statique (qui a l'extension `.lib` sous Windows). Pour configurer le type de projet produit par le code, il faut le configurer dans les propriétés du projet (clic droit sur le projet puis « Propriétés » ; vous modifierez alors le « *Type de configuration* » en sélectionnant « *Bibliothèque statique (.lib)* »)

Ajoutez un fichier C qui contiendra le code de la bibliothèque. Pour cela, faites un clic droit sur le dossier « *Fichiers sources* » du projet puis « *Ajouter* / *Nouvel élément* ». « *Visual C++* » doit être sélectionné sur la gauche et sélectionnez « *Fichier C++* ». Vous nommerez le fichier `printStop.c`.

Attention ! Vous donnerez bien l'extension `.c` et non pas `.cpp` à votre fichier. En choisissant .c, l'environnement comprendra que vous souhaitez faire du code C et non pas C++. Si vous conservez `.cpp`, l'IDE utilisera le compilateur C++ (au lieu de C) et vous n'arriverez pas à une solution fonctionnelle.

Ecrire le code source `printStop.c` ne contenant qu'une méthode `void PrintStop(char* msg)` qui affiche le message `msg`. Etant donné que les bibliothèques Win32 respectent le standard C ANSI et donc une partie de la norme POSIX, vous pourrez utiliser `printf()` contenu dans `stdio.h`. La fonction `PrintStop` se terminera par un appel à `getchar` qui attendra l'envoi d'un caractère quelconque.

Afin que votre méthode fasse partie de l'API de votre bibliothèque, il vous faut exporter cette procédure :

```
_declspec(dllexport) void PrintStop(char*);
```

Générer la solution et vérifier que les fichiers nécessaires à l'utilisation de la bibliothèque ont été générés dans le dossier `Debug` de votre solution (et donc en particulier le fichier `.lib`).

2.2 Utilisation de la bibliothèque statique

Exercice n°2:

Ajouter un nouveau projet à la solution que vous nommerez `Exe_DynLink` (créez à nouveau ce projet à partir du modèle « *Projet vide* »). Vous ajouterez un fichier `main.c` à ce nouveau projet (attention à nouveau à bien mettre l'extension `.c` et non pas `.cpp` !). Ce programme utilisera la bibliothèque que vous avez réalisée à l'exercice précédent. Pour cela, vous ajouterez une référence au projet « Bibliotheques » pour le projet `Exe_DynLink`.
Vérifier le bon fonctionnement de ce programme en l'exécutant. Rendez-vous dans le dossier Debug de votre solution. Identifiez les fichiers qui ont été produits par la compilation. Effacez tous les fichiers sauf le programme `Exe_DynLink.exe` pour vérifier que le programme est fonctionnel dans le fichier Bibliotheques.lib et donc que la bibliothèque a bien été incluse dans le programme à l'édition de liens.

2.3 Bibliothèque partagé et édition de lien dynamique

Exercice n°3:

Modifiez le type de bibliothèque produite en sélection maintenant le type de configuration « *Bibliothèques dynamique (.dll)* » dans le projet « Bibliotheques ». Nettoyez la solution et regénérez celle-ci. Rendez-vous dans le dossier Debug de votre solution. Supprimez les fichiers inutiles afin de vérifier ceux vraiment nécessaires à l'exécution.

TD n° 2 bis
Bibliothèques Statiques et Dynamiques sous Win32

3 Bibliothèque dynamique avec chargement dynamique

Exercice n°4:

Créez un nouveau projet à partir du modèle « Projet Vide » que vous nommerez Exe_DynLoad. Ajoutez un fichier main.c à votre projet. Ce programme devra charger dynamiquement à l'exécution la bibliothèque partagée que vous avez créée précédemment. Vous utiliserez pour cela les fonctions l'API Win32 : `LoadLibrary`, `GetProcAddress`, `FreeLibrary`.

Pour vous aider, voici un code source commenté qui permet d'utiliser une fonction `int convert(dollar int)` de la bibliothèque `D:\Temp\Monnaie.dll`.

```
#include <Windows.h>
#include <stdio.h>

// décrire le prototype de la procédure : proto_convert est le type défini par typedef,
// c'est un pointeur sur fonction int (int).
typedef int(* proto_convert) (int);

int main()
{
    proto_convert convert;

    // Attention : le chemin du fichier est en Unicode si votre projet est configuré
    //             comme tel, la chaine doit alors être L" "
    // Attention : pas de \ dans le chemin, sinon c'est un caractère d'échappement
    void * hinstDLL = LoadLibrary("D:/Temp/Monnaie.dll");
    if (!hinstDLL) // Erreur lors du chargement de la bibliothèque ?
           printf("Impossible de charger la bibliothèque.");
    else          // On récupère l'adresse de la fonction
           convert = (proto_convert) GetProcAddress(hinstDLL, "convert");
    int a = 5;
    convert(a);
    FreeLibrary(hinstDLL);

    //convert(a); // qu'arrive-t'il?
}
```

4 Conclusion

Exercice n°5:

Que pouvez-vous constater comme différences dans la création et l'utilisation des bibliothèques sous Linux et sous Windows (ponts communs et différences entre les deux systèmes).

5 Exercice complémentaire non obligatoire

Exercice n°6:

Essayez de créer l'exécutable tri.exe qui réalise le chargement dynamique de bibliothèques partagées tri_xxx.dll tel que vous l'avez réalisé sous Linux, mais cette fois-ci sous Windows.