

Compilation Delphi en ligne de commande


par dieamondo ([\[-Aka-\] Prod](#))

Date de publication : 08/11/2007

Dernière mise à jour :

Cet article explique comment compiler un programme Delphi en ligne de commande sur un poste dont aucune version de Delphi n'est déjà installée.

- I - Préambule
- II - Préparation des dossiers
- III - Les fichiers binaires
- IV - Configuration du compilateur
- V - Gestion des librairies
 - V-A - Les librairies Delphi
 - V-B - Les librairies tierce partie
 - V-B-1 - Méthode manuelle (recommandée)
 - V-B-2 - Méthode automatique
- VI - Compilation du projet en ligne de commande
- VII - Aller plus loin

 *Ce tutoriel explique simplement une technique de compilation et la manipulation de certains outils en ligne de commande. Avant tout usage dans un cadre professionnel ou personnel, veuillez penser à vérifier que cette utilisation est légale et autorisée par la licence Borland ou CodeGear que vous possédez.*

I - Préambule

Compiler un projet Delphi sans avoir installé l'EDI complet, c'est très simple !

Mais vous me direz, à quoi ça sert ? Cela dépend tout simplement du projet que vous avez en tête. Cela peut être intéressant pour créer des applications dynamiques, des patches, des DLLs, ... ou tout simplement automatiser certains processus de compilation complexes. Cette technique peut être intéressante pour effectuer également certaines compilations directement chez le client, suite à une intervention technique.

Ce tutoriel a été réalisé avec Delphi7 et Delphi2007. Pour d'autres versions, il faudra sûrement adapter certaines étapes, notamment au sujet de certains noms de fichiers indexés par la version de l'EDI à laquelle ils se rapportent.

II - Préparation des dossiers

Avant toute chose, nous allons créer un répertoire qui va héberger notre projet. Nommons le par exemple *ProjectCMD*.

Ensuite nous allons organiser nos fichiers en 3 répertoires :

- **Binaires** : Contiendra les fichiers EXE, DLL et CFG de Delphi, nécessaires a la compilation.
- **Librairies** : Contiendra les fichiers DCU des librairies utilisées par notre application.
- **Sources** : Contiendra les sources de votre projet Delphi.

III - Les fichiers binaires

Il va nous falloir quelques fichiers (4 au total) nécessaires à la compilation d'un projet Delphi :

- **DCC32.exe** : Delphi Pascal Compiler.
- **DCC32.cfg** : Delphi Pascal Compiler Configuration.
- **Inkdfm70.dll** : Liaison des .DFM par rapport aux fiches.
- **rlink32.dll** : Support des ressources 32bits.

Vous trouverez ces fichiers dans le répertoire d'installation de Delphi : *{Delphi}\Bin*.

Copiez les dans le répertoire **ProjectCMD\Binaires**.



Pour les autres versions de Delphi, la liste de ces fichiers peut être différente, tout comme leur nom. Par exemple le fichier Inkdfm70.dll est indexé par le numéro de version de l'IDE auquel ils se rapporte : 70 pour Delphi 7.

IV - Configuration du compilateur

Le compilateur en ligne de commande de Delphi possède toute une série d'options pour paramétrer son exécution. En voici la liste exhaustive :

```
C:\Program Files\Borland\Delphi7\Bin>dcc32
Borland Delphi Version 15.0
Copyright (c) 1983,2002 Borland Software Corporation

Syntaxe : dcc32 [options] fichier [options]

-A<unité>=<alias> = Alias d'unité      -LU<paquet> = Utiliser paquet
-B = Construire toutes les unités     -M = Construire unités modifiées
-CC = Cible console                   -N<chemin> = destination DCU
-CG = Cible GUI                       -O<chemins> = Répertoires objets
-D<syms> = Définir conditionnels      -P = Chercher aussi fichiers 8.3
-E<chemin> = Répertoire dest. EXE     -Q = Compilation silencieuse
-F<offset> = Chercher erreur          -R<chemins> = Répertoires ressources
-GD = Fichier map détaillé            -U<chemins> = Répertoires unités
-GP = Fichier map avec publics        -V = Infos débogage dans EXE
-GS = Fichier map avec segments       -VR = Générer débogage distant (RSM)
-H = Afficher les conseils            -W = Afficher les avertissements
-I<paths> = Répertoires d'inclusion     -Z = Générer DCPs
-J = Générer fichier .obj             -$<dir> = Directive de compilation
-JP = Générer fichier .obj C++       --help = Affiche cet écran d'aide
-K<addr> = Adresse de base image      --version = Affiche le nom et la version

Options du compilateur : -$<lettre><état> (voir défauts ci-dessous)
A8 Champs enregistrements alignés    P+ Params chaîne ouverte
B- Evaluation booléenne complète     Q- Vérification débordement entier
C+ Evaluer assertions à l'exécution  R- Vérification étendue
D+ Informations de débogage           T- Opérateur @ typé
G+ Utiliser réf. données importées   U- Division Pentium(tm) sûre
H+ Utiliser chaînes longues par défaut V+ Chaîne variables strictes
I- Vérification E/S                  W- Générer cadre de pile
J- Consts structurées en écriture     X+ Syntaxe étendue
L+ Symboles débogage locaux          Y+ Info référence symbole
M- Info type à l'exécution           Z1 Taille mini types énumérés
O+ Optimisations

C:\Program Files\Borland\Delphi7\Bin>
```

A l'usage, la ligne de commande peut devenir très complexe, compte tenu de la multitude des options et leur taille. Rien que pour l'option qui nous intéresse : `-U<chemins> = Répertoires unités`, la ligne de commande peut prendre facilement 3 à 4 lignes, pour peu que l'application utilise plusieurs librairies

Pour simplifier cette ligne de commande, Borland à créé un fichier de configuration `dcc32.cfg` qui nous permet de définir certaines options par défaut. On note dans ce fichier une série d'options que le compilateur ajoutera à la suite de celles transmises en ligne de commande, avant de procéder à la compilation.

Si on regarde ce fichier, on peut noter la ligne suivante :

```
-u"C:\Program Files\Borland\Delphi7\lib";"C:\Program Files\Borland\Delphi7\lib\Obj"
```

D'après la liste affiché plus haut, l'option `-u` définit la liste des répertoires contenant les unités. En un mot la *librairie*.

Nous allons donc modifier ces chemins afin qu'il pointe vers notre répertoire **ProjectCMD\Librairies**, qui contiendra l'ensemble des librairies utilisées. Pour faciliter la migration de notre application d'un poste à l'autre, il est préférable d'utiliser les chemins relatifs, ce qui nous donne ceci :

```
-u..\Librairies";
```

Afin de faciliter la maintenance de notre application, il est conseillé de regrouper les différentes librairies en sous répertoires distincts. Par exemple :

- **Delphi** : Librairie fournie par Delphi
- **JCL** et **JVCL** : Librairies JEDI
- **TMS** : Librairies TMS Software
- Etc.


Dans ce cas, le fichier de configuration ressemblera à ceci :

```
-u..\Librairies\Delphi";..\Libraires\JCL";..\Libraires\JVCL";..\Libraires\TMS";
```

Certains chemins peuvent volontairement être omis pour être transmis dans la ligne de commande. Cela permet de spécialiser certaines compilations et d'éviter ce que j'appelle une *collision* entre 2 librairies. Par exemple, compiler une application en s'appuyant sur d'anciennes versions de certaines librairies.

De plus, ces chemins peuvent référencer des chemins précis sur le disque dur, externes à notre projet (*G:\MesLibs\MaLibStandard*). Dans ce cas, il sera difficile voir impossible de changer de poste, sans modifier le fichier de configuration, et copier l'intégralité de ces librairies.

Notre conseil est donc de réunir au sein de notre projet l'ensemble des librairies sur lesquelles notre application s'appuie, et de définir des chemins relatifs. Ainsi, une simple copie de notre répertoire "ProjectCMD" suffit à compiler notre application sur n'importe quel poste.

 *Certaines options, spécifiques à un projet, peuvent être définies directement dans les sources du projet : Project1.cfg (ou project1 est le nom de votre projet), situées juste à côté de votre fichier DPR. Cela permet de définir des options différentes au divers projets d'une solution : Exe, Dlls, etc.*

V - Gestion des librairies

Pour qu'un projet se compile, il lui faut avoir accès à l'ensemble des unités utilisés.

Lister toutes les unités nommées dans les diverses sections *uses* de votre projet ne suffit pas. En effet, certaines unités s'appuient sur d'autres de manière transparente.

Pour exemple, un projet graphique vierge (*Project1.dpr*) utilise *Unit1* qui elle-même utilise *Windows*, *Messages*, *SysUtils*, *Variants*, *Classes*, *Graphics*, *Controls*, *Forms* et *Dialogs*, qui ne sont pas directement déclarées dans le source du projet (*Project1.dpr*).

Le moyen le plus simple d'arriver à toutes les lister est de s'appuyer sur le compilateur Delphi, qui échec après échec, citera les unités manquantes. Ce sont ces dernières qu'il faudra ajouter à notre répertoire *ProjectCMD\Librairies* pour compléter la compilation.

V-A - Les librairies Delphi

Voici une liste des unités nécessaires à la compilation d'une application vierge Delphi. Cette liste peut ensuite varier en fonction des différentes versions de Delphi et de la gamme de composants utilisés. Toutefois elle offre une bonne base de départ.

L'ensemble de ces fichiers se trouve dans le répertoire *{Delphi}\Lib* et devra donc être copié dans notre répertoire *ProjectCMD\Librairies* ou bien *ProjectCMD\Librairies\Delphi* si on utilise le système de séparation des librairies conseillé.

Projet Delphi 7 [GUI] (66 Fichiers)

```
ActiveX.dcu;ActnList.dcu;Buttons.dcu;buttons.res;Classes.dcu;Clipbrd.dcu;ComCtrls.dcu;CommCtrl.dcu;CommDlg.dcu;
ComObj.dcu;ComStrs.dcu;Consts.dcu;Contnrs.dcu;Controls.dcu;controls.res;Dialogs.dcu;Dlgs.dcu;ExtActns.dcu;
ExtCtrls.dcu;ExtDlgs.dcu;extdlgs.res

FileCtrl.dcu;filectrl.res;FlatSB.dcu;Forms.dcu;Graphics.dcu;HelpIntfs.dcu;ImgList.dcu;Imm.dcu;IniFiles.dcu;
ListActns.dcu;Mapi.dcu;Math.dcu;Menus.dcu;Messages.dcu;MMSystem.dcu;MultiMon.dcu;PenWin.dcu;Printers.dcu;
Registry.dcu;RegStr.dcu;RichEdit.dcu

RTLConsts.dcu;ShellAPI.dcu;ShlObj.dcu;StdActns.dcu;StdCtrls.dcu;StdVCL.dcu;StrUtils.dcu;SyncObjs.dcu;
SysConst.dcu;SysInit.dcu;System.dcu;SysUtils.dcu;Themes.dcu;ToolWin.dcu;Types.dcu;TypInfo.dcu;UrlMon.dcu;
UxTheme.dcu;Variants.dcu;VarUtils.dcu;Windows.dcu

WinHelpViewer.dcu;WinInet.dcu;WinSpool.dcu
```

Projet Delphi 7 [Console] (6 fichiers)

```
SysConst.dcu;SysInit.dcu;System.dcu;SysUtils.dcu;Types.dcu;Windows.dcu
```

Projet Delphi 2007 [GUI] (52 Fichiers)

```
ActiveX.dcu;ActnList.dcu;Classes.dcu;Clipbrd.dcu;CommCtrl.dcu;CommDlg.dcu;Consts.dcu;Contnrs.dcu;Controls.dcu;
controls.res;Dialogs.dcu;Dlgs.dcu;DwmApi.dcu;ExtCtrls.dcu;FlatSB.dcu;Forms.dcu;Graphics.dcu;GraphUtil.dcu;
HelpIntfs.dcu;ImageHlp.dcu;ImgList.dcu;

Imm.dcu;IniFiles.dcu;Math.dcu;Menus.dcu;Messages.dcu;MultiMon.dcu;Printers.dcu;Registry.dcu;RegStr.dcu;
RTLConsts.dcu;ShellAPI.dcu;ShlObj.dcu;StdActns.dcu;StdCtrls.dcu;StrUtils.dcu;SyncObjs.dcu;SysConst.dcu;
SysInit.dcu;System.dcu;SysUtils.dcu;Themes.dcu;
```

Projet Delphi 2007 [GUI] (52 Fichiers)

```
Types.dcu;TypInfo.dcu;UrlMon.dcu;UxTheme.dcu;Variants.dcu;VarUtils.dcu;WideStrUtils.dcu;Windows.dcu;  
WinInet.dcu;WinSpool.dcu;
```

Projet Delphi 2007 [Console] (7 Fichier)

```
ImageHlp.dcu;SysConst.dcu;SysInit.dcu;System.dcu;SysUtils.dcu;Types.dcu;Windows.dcu
```

V-B - Les librairies tierce partie

Si vous utilisez des composants ou des librairies tierce partie dans votre projet, il vous faudra aussi leurs unités.

Je vous propose 2 solutions pour trouver ces unités :


V-B-1 - Méthode manuelle (recommandée)

Cette méthode consiste à rechercher et regrouper l'ensemble des unités utilisées dans un des sous répertoires de *ProjectCMD\Librairies*. Après ajout de ces chemins au fichier de configuration du compilateur, ce dernier trouvera les fichiers nécessaires et compilera avec succès.

V-B-2 - Méthode automatique

Delphi vous permet de regrouper, à la compilation, les fichiers DCU des unités d'un projet dans un répertoire de votre choix. Si ce n'est pas déjà fait, dirigez-vous dans l'EDI vers *Projet > Options > onglet Répertoires/Conditions* puis, remplissez le champ *Destination de l'unité*.

Un tri parmi les fichiers DCU copiés dans ce répertoire après compilation permet de compléter notre répertoire *Librairies*.

 *Quelle que soit la méthode utilisée, il est conseillé de bien vérifier l'ensemble des options du projet et du compilateur, afin que la compilation utilise les bons fichiers. Bien souvent, le compilateur utilise un ancien fichier soit en utilisant un vieux chemin d'accès soit parce que le développeur a mal géré les règles de priorités de recherches du compilateur*

VI - Compilation du projet en ligne de commande

Pour finir, il suffit de lancer le compilateur en ligne de commande pour compiler votre projet.

Pour cela :

- Ajoutez les fichiers source de votre projet dans notre dossier **Sources**.
- Lancer une invite de commande (*Demarrer > Executer > cmd*)).
- Se positionner dans votre dossier (*ProjectCMD*). - Lancer la compilation avec la commande : `.\Binaires\dcc32 -b .\Sources\MonProjet.dpr`.

En cas de problème, lisez bien ce que vous affiche le compilateur. Par exemple en cas de DCU manquant le compilateur affichera le texte suivant : *Fatal : File not found : 'Monfichier.dcu'*. Si tel était le cas, faites une recherche de ce fichier et placez-le dans un des répertoires de *ProjectCMD\Librairies*.

Je vous conseille d'écrire un fichier *batch* nommé par exemple *compile.bat* afin d'automatiser la compilation, que l'on sauvegardera dans le répertoire racine de notre projet : *ProjectCMD*.

Fichier batch de compilation

```
@ECHO OFF
.\Binaires\dcc32 -b .\Sources\MonProjet.dpr
```

VII - Aller plus loin

Cet article donne les bases de la compilation d'un projet en ligne de commande. Il est possible d'affiner sa compilation en utilisant toute la gamme des options présentées dans le chapitre IV : *Configuration du compilateur*.

Ces options peuvent être utilisées en ligne de commande, ou bien spécifiées dans les différents fichiers de configuration (projet et compilateur), comme évoqué plus haut.

Si l'on est amené à compiler plusieurs projets, il est possible de séparer la partie *Binaires* et *Librairies* dans un répertoire nommé *DelphiCMD*, de la partie *Projets*. En faisant pointer la variable d'environnement *PATH* du système d'exploitation vers le fichier contenant les binaires (*DelphiCMD\Binaires*). On obtient ainsi une version minimale de Delphi, excluant l'IDE.

Je terminerais par la présentation brève de quelques binaires supplémentaires qui peuvent agrandir les possibilités de notre système de compilation en ligne de commande :

- **brcc32.exe** : Compilateur de ressources en ligne de commande. Il permet de générer des fichiers de ressources compilés *.RES* à partir de fichiers source *.RC*. Combiné avec *dcc32.exe* il est l'outil idéal pour compiler à *la volée* des DLLs de ressources...

- **tregrsv.exe** : Utile pour enregistrer des DLL COM ou ActiveX dans la base de registre. C'est l'équivalent de *regsvr32* livré par Windows ...

Tutoriel rédigé par **[-Akado-]** pour Developpez.com, corrigé et complété par **Clorish**.

(Merci à ma femme qui ma laissé le temps d'écrire ce tutorial)

