

Tutoriel pour Borland C++ Compiler 5.5

par [Pascal Coudert](#)

Date de publication : 09/12/2001

Dernière mise à jour :

J'ai trouvé sur le site Borland le compilateur gratuit Borland C++ 5.5 permettant de réaliser des programmes pour DOS (fenêtre DOS sous Windows 32 bits pour être plus précis) et Windows et je vais essayer de vous faire partager mes découvertes concernant l'utilisation de ce compilateur. Si vous avez d'autres informations intéressantes, n'hésitez pas à m'en faire part. Je pense que la première chose à faire est de télécharger le compilateur qui en fait est une version réduite de Borland C++ builder. Il ne contient pas d'éditeur de code ou environnement de développement intégré IDE comme Borland C++ Builder ou les outils de développement visuels mais il est parfaitement opérationnel pour réaliser de grandes applications et donner matière à réfléchir. En ce qui me concerne, je préfère maîtriser toutes les lignes de code de mes programmes et les outils soit disant magiques de développement visuel d'applications me font peur

Télécharger le compilateur gratuit Borland C++ 5.5
Installer le compilateur gratuit Borland C++ 5.5
Tester l'installation du compilateur gratuit Borland C++ 5.5
 Programme de test fonctionnant sous DOS
 Programme de test fonctionnant sous Windows 9x
Se procurer de l'aide et utiliser les fonctions API
Construire la fenêtre principale d'une application Windows
Ajouter des ressources (icônes, menus, boutons etc...) à une fenêtre
 En route pour compiler testres.c
 VI-B - En route pour compiler testres.rc
 En route pour l'édition de liens (ou linkage)
Réaliser un fichier batch de projet

Télécharger le compilateur gratuit Borland C++ 5.5

Pour télécharger le compilateur gratuit **Borland C++ 5.5** vous devez vous connecter sur le site <http://www.inprise.fr/download/compilateurs/> et suivre les indications proposées (vous devrez d'abord vous enregistrer gratuitement pour obtenir des codes d'accès qui vous permettront de procéder au téléchargement). Le nom de fichier du compilateur gratuit **Borland C++ 5.5** est **FreeCommandLineTools.exe** et il nécessite **8.54Mo** d'espace sur votre disque dur.

Installer le compilateur gratuit Borland C++ 5.5

Pour installer le compilateur gratuit il vous suffit de cliquer sur le nom du fichier (**FreeCommandLineTools.exe**) et de laisser le programme d'installation utiliser les options par défaut. Tous les fichiers seront donc copiés dans le dossier **c:\Borland\Bcc5.5**.

Etant donné que ce compilateur est livré sans Environnement de développement Intégré, vous devrez le faire fonctionner sous DOS dans un premier temps ou plutôt à partir de Windows dans une fenêtre DOS. De plus, comme vous placerez tous les fichiers sources de l'application que vous souhaitez réaliser dans un seul dossier, il vous sera plus facile de lancer le compilateur **Bcc32.exe** à partir de ce dossier (on ne l'écrit pas pour rien...) en tapant la commande **bcc32**. Pour que le compilateur puisse être lancé de cette manière, vous devez ajouter à votre fichier autoexec.bat la ligne suivante : **path=c:\Borland\Bcc55\bin** et redémarrer votre ordinateur.

Si vous tapez ensuite **bcc32** sous DOS ou plutôt dans une fenêtre Dos (on ne va pas le répéter à chaque fois...), le compilateur **Borland C++ 5.5** affiche ses options de ligne de commande :

```

Microsoft(R) Windows 95
(C)Copyright Microsoft Corp 1981-1996.

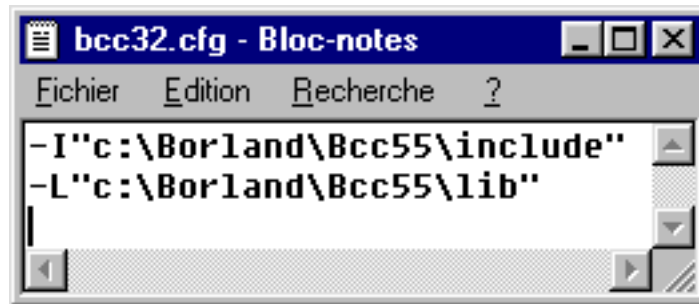
C:\WINDOWS>bcc32
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] [ file(s) ]
-3 * 80386 Instructions -4 * = default; -x- = turn switch x off
-5 Pentium Instructions -6 Pentium Pro Instructions
-Ax Disable extensions -B Compile via assembly
-C Allow nested comments -Dxxx Define macro
-Exxx Alternate Assembler name -Hxxx Use pre-compiled headers
-Ixxx Include files directory -K Default char is unsigned
-Lxxx Libraries directory -M Generate link map
-N Check stack overflow -Ox Optimizations
-P Force C++ compile -R Produce browser info
-RT * Generate RTL -S Produce assembly output
-Ixxx Set assembler option -Uxxx Undefine macro
-U Virtual table control -X Suppress autodep. output
-aN Align on N bytes -b * Treat enums as integers
-c Compile only -d Merge duplicate strings
-exxx Executable file name -fxx Floating point options
-gN Stop after N warnings -iN Max. identifier length
-jN Stop after N errors -k * Standard stack frame
-lx Set linker option -nxxx Output file directory
-oxxx Object file name -p Pascal calls
-tWxxx Create Windows app -u * Underscores on externs
-v Source level debugging -wxxx Warning control
-xxxx Exception handling -y Produce line number info
-zxxx Set segment names

C:\WINDOWS>
  
```

Pour ne pas avoir besoin de taper les commandes nécessaires à chaque compilation (à la suite de **Bcc32**) , il est prévu que le compilateur aille lire le fichier **Bcc32.cfg** qu'il devra normalement trouver dans le dossier **c:\Borland\Bcc5.5\bin**. Par conséquent, le mieux est de le configurer tout de suite en y inscrivant les lignes suivantes :

-I"c:\Borland\Bcc55\include" : Indique au compilateur qu'il doit aller chercher dans le dossier **c:\Borland\Bcc55\include** les fichiers d'inclusion contenant les définitions des fonctions standard que vous utilisez dans votre programme et que vous lui mentionnez par la directive **#include**.

-L"c:\Borland\Bcc55\lib" : Indique au compilateur qu'il doit aller chercher dans le dossier **c:\Borland\Bcc55\lib** les fichiers **.lib** contenant les informations concernant les fonctions des dll correspondantes.



Etant donné que le compilateur lancera le programme d'édition de liens **ilink32.exe** pour générer le fichier exécutable de votre programme, il est aussi souhaitable de configurer le fichier **ilink32.cfg** en y inscrivant la ligne suivante :

-L\"c:\Borland\Bcc55\lib\" : Indique à l'éditeur de liens qu'il doit aller chercher dans le dossier **c:\Borland\Bcc55\lib** les fichiers **.lib** contenant les informations pour lier à votre programme les fonctions des **dll** correspondantes.



En tapant **ilink32** sous DOS... on obtient les options de ligne de commande de l'éditeur de liens **ilink32.exe** :

```

Microsoft(R) Windows 95
(C) Copyright Microsoft Corp 1981-1996.

C:\WINDOWS>ilink32
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
Syntax: ILINK32 objfiles, exefile, mapfile, libfiles, deffile, resfiles
@xxxx indicates use response file xxxx
General Options:
-C Clear state before linking
-wxxx Warning control
-Enn Max number of errors
-r Verbose linking
-g Suppress banner
-c Case sensitive linking
-u Full debug information
-Gn No state files
-Gi Generate import library
-GD Generate .DRC file
Map File Control:
-M Map with mangled names
-m Map file with publics
-s Detailed segment map
-x No map
Paths:
-l Intermediate output dir
-L Specify library search paths
-j Specify object search paths
Image Control:
-d Delay load a .DLL
-Af:nnnn Specify file alignment
-Ao:nnnn Specify object alignment
-ax Specify application type
-b:xxxx Specify image base addr
-Ixx Specify output file type
-H:xxxx Specify heap reserve size
-Hc:xxxx Specify heap commit size
-S:xxxx Specify stack reserve size
-Sc:xxxx Specify stack commit size
-Ud.d Specify Windows version
-Dstring Set image description
-Ud.d Specify subsystem version
-Ud.d Specify image user version
-GC Specify image comment string
-GF Set image flags
-GI Static package
-Gpd Design time only package
-Gpr Runtime only package
-GS Set section flags
-ET East TLS
-Gz Do image checksum
-Rr Replace resources
  
```


Tester l'installation du compilateur gratuit Borland C++ 5.5

Pour tester si le compilateur **Borland C++ 5.5** est correctement installé sur votre ordinateur, je vous propose de lui faire réaliser un petit exécutable de test fonctionnant sous DOS et un autre fonctionnant sous **Windows 9x**.

Programme de test fonctionnant sous DOS



Programme de test fonctionnant sous Windows 9x



Se procurer de l'aide et utiliser les fonctions API

En fait, toutes les fonctions de base qui permettent de développer un programme pour Windows sont regroupées dans un ensemble appelé **API** (**A**pplication **P**rogramming **I**nterface) ces fonctions ont leurs points d'entrées dans le fichier **windows.h** situé dans le dossier **c:\Borland\Bcc55\include**. C'est la raison pour laquelle on place la ligne **#include <windows.h>** au début du code du programme. Si on est amené à utiliser des fonctions écrites dans un autre fichier du dossier **c:\Borland\Bcc55\include** alors il faudra ajouter au début du programme la ligne **#include <nom du fichier>** correspondante. Si vous préférez placer le fichier dans le même dossier que vos autres sources alors vous écrirez **#include 'nom du fichier'**.

La documentation des fonctions **API** et des ressources standard d'une application **Windows 32 bits** se trouve dans le fichier **win32.hlp** fourni par **Microsoft**. On le trouve avec la plupart des compilateurs (payants). Vous pouvez aussi utiliser les moteurs de recherche du Web pour le trouver en téléchargement sur certains sites. Pour ma part j'ai utilisé le moteur [Google](#) qui m'a sorti le lien suivant :

<http://www.inprise.com/devsupport/delphi/downloads/>

Sur la page correspondante vous trouverez une entrée au fichier **win32.zip** qui contient le fichier **win32.hlp** :

<http://www.inprise.com/devsupport/delphi/downloads/win32.zip>(7.59 Mo)

Vous pouvez aussi vous aider des fichiers d'aide de l'**API Windows CE** pour une première approche des fonctions dont vous avez besoin avant d'approfondir la recherche dans **win32.hlp** ou ailleurs.

Les fichiers d'aide de l'API Windows CE se trouvent en téléchargement sur le site de Microsoft :

<http://msdn.microsoft.com/msdn-files/027/001/460/wince212docs.exe> (5.05 Mo)

Une autre source importante de renseignements se trouve dans le kit **platform SDK de Microsoft** ainsi que **MSDN** mais bon, pour l'instant nous n'en avons pas vraiment besoin.

Construire la fenêtre principale d'une application Windows

Voici le code constituant le squelette de base d'une **application Windows 32 bits** que nous allons réaliser avec le compilateur gratuit **Borland C++ 5.5** et que nous pourrions modifier par la suite pour aller un peu plus loin. Le **C++** étant une extension du **C** et que personnellement je trouve que le **C** est plus simple que le **C++** et qu'ensuite le compilateur gratuit **Borland C++ 5.5** compile très bien le **C**, nous allons travailler en **C**. De toutes façons je ne comprends pas le **C++** et je n'ai pas envie de le comprendre parce que je trouve que je suis suffisamment organisé pour ne pas avoir besoin d'être guidé artificiellement par le langage de programmation lui-même et ses nombreuses contraintes. De plus les fonctions **API** sont écrites en **C** et les exemples qui s'y rapportent aussi.

```
#include <windows.h>

//déclarations de variables
LPSTR lpszAppName="Ma première fenêtre Windows";
HINSTANCE hInst;
HWND hWnd;

//Prototypes de fonctions
LONG WINAPI WndProc( HWND hWnd, UINT uMsg, WPARAM wParam,LPARAM lParam );

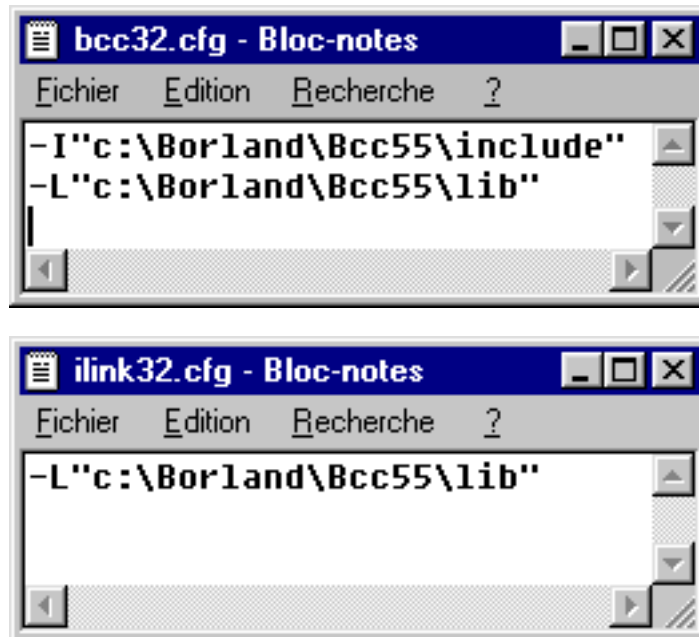
//Fonctions
int APIENTRY WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
{
    MSG msg;
    WNDCLASS cls;

    // Enregistrer la classe fenêtre de l'application principale.
    cls.hInstance = hInstance;
    cls.lpszMenuName = lpszAppName;
    cls.lpszClassName = lpszAppName;
    cls.hIcon = LoadIcon(NULL, IDI_EXCLAMATION);
    cls.hCursor = LoadCursor(NULL, IDC_ARROW);
    cls.hbrBackground =(HBRUSH)(COLOR_WINDOW+1);
    cls.style = CS_VREDRAW | CS_HREDRAW;
    cls.lpfnWndProc = (WNDPROC)WndProc;
    cls.cbWndExtra = 0;
    cls.cbClsExtra = 0;
    if ( !RegisterClass( &cls ) )
        return( FALSE );
    hInst = hInstance;

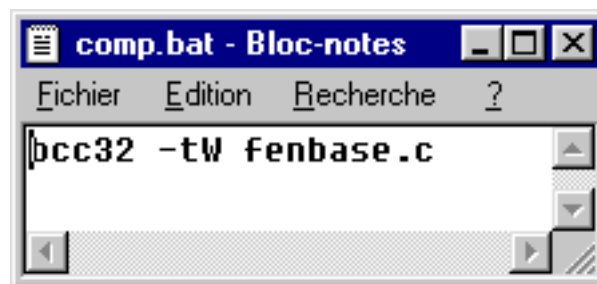
    // créer l'application principale
    hWnd = CreateWindow (lpszAppName, lpszAppName, WS_OVERLAPPEDWINDOW, 50, 50, 640, 470, NULL, NULL,
hInst, NULL);
    if ( !hWnd )
        return( FALSE );
    ShowWindow( hWnd, nCmdShow );
    UpdateWindow( hWnd );
}
```

```
while( GetMessage( &msg, NULL, 0, 0 ) )
{
    TranslateMessage( &msg );
    DispatchMessage( &msg );
}
return( msg.wParam );
}
LONG WINAPI WndProc( HWND hWnd, UINT uMsg, WPARAM wParam , LPARAM lParam )
{
    switch(uMsg)
    {
        case WM_CLOSE :
        {
            DestroyWindow( hWnd );
        };
        break;
        case WM_DESTROY :
        {
            PostQuitMessage(0);
        };
        break;
        case WM_QUERYENDSESSION :
        {
            DestroyWindow( hWnd );
        };
        break;
        default :
            return DefWindowProc( hWnd, uMsg, wParam, lParam );
    };
    return 0;
}
```

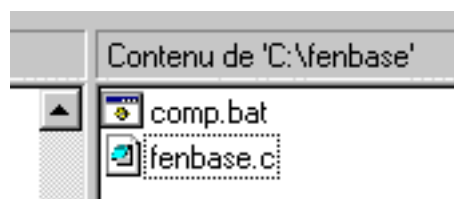
Après avoir écrit le listing ci-dessus avec votre éditeur de texte et l'avoir enregistré sous le nom **fenbase.c** et placé dans le dossier **c:\fenbase** de votre disque dur. Bon, vous pouvez aussi le télécharger directement ici : [fenbase.c](#) . On remarquera que l'extension du fichier n'est plus **.cpp** mais **.c** . Ceci indiquera au compilateur ou aux lecteurs (c'est plus probable) que le programme est écrit en **langage C** (bon, si on avait laissé **.cpp** ça n'aurait pas changé grand chose comme dirait l'autre puisque le **C** est un sous ensemble du **C++**. Peut-être au niveau du temps de compilation, je laisse ce calcul aux spécialistes.) bref maintenant il faut écrire le **batch** pour lancer le compilateur avec les commandes nécessaires sans avoir besoin de tapoter du clavier dans le DOS à chaque fois. Certains préfèrent utiliser les fichiers **make** mais pour ma part je trouve que c'est trop galère. Récapitulatif des fichiers de configuration devant être placés dans **c:\Borland\Bcc5.5\bin**:



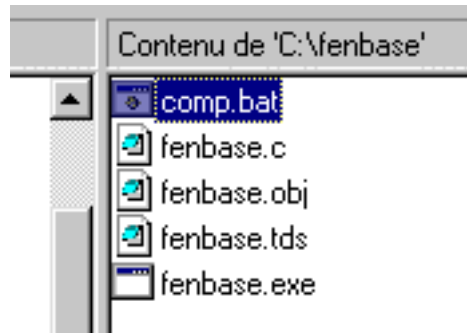
Le fichier **batch** sera **comp.bat** et placé dans **c:\fenbase**. Son contenu sera identique à ce que nous avons tapé sous DOS pour réaliser le programme de test Windows mais avec **fenbase.c** à la place de **hello**.



Par conséquent le contenu de **c:\fenbase** est le suivant :



Et après un double clic sur **comp.bat**, quelques secondes de patience et un clic sur la croix de la fenêtre DOS pour la fermer lorsque la compilation est terminée on obtient :



Miracle ! voici notre petit [fenbase.exe](#) sur lequel on peut double cliquer pour voir s'afficher notre fenêtre de base :



Bon je l'ai volontairement un peu réduite il faut bien l'avouer.

Ajouter des ressources (icônes, menus, boutons etc...) à une fenêtre

Pour l'instant nous n'avons pas encore de fenêtre avec menus, boutons etc... mais c'est en bonne voie. à la fin de ce paragraphe nous aurons réussi à créer une fenêtre qui se transforme en icône lorsqu'elle est réduite c'est à dire que son icône (non standard fournie par Windows) apparaîtra dans l'explorateur ou comme raccourci qu'on voudra bien placer sur le bureau etc.... mais surtout on aura appris à utiliser un fichier de ressources dans lequel on pourra ajouter toutes les ressources qu'on voudra pour notre application.

Bon pour notre étude on dira que notre petit bout de programme s'appellera **testres.exe** et que tous les fichiers que nous allons créer pour le développer, en particulier le code source (**testres.c**) de la fenêtre seront placés dans le dossier **c:\testres** alors à vous de jouer pour la création du dossier.

Une icône est une ressource et elle doit être mentionnée dans un fichier de ressources (**testres.rc**). C'est aussi un dessin (**testres.ico**) réalisé avec un programme de dessin d'icônes. Ce fichier de ressources devra contenir les éléments suivants :

```
MAINICON ICON LOADONCALL MOVEABLE DISCARDABLE testres.ico
```

Vous trouverez le descriptif de la ressource **ICON** dans le fichier **rc.hlp** qui est lui-même dans le fichier **b5ms.zip** sur le site ftp de **inprise** : <ftp://ftp.inprise.com/pub/bcppbuilder/techpubs/b5ms.zip> mais ce fichier pèse **17Mo**. Pour en extraire un **rc.hlp** de 237 Ko ça fait beaucoup. Je vous propose alors de télécharger une version plus ancienne directement [ici](#) du fichier **rc.hlp**. Plus ancienne mais plus légère (132 Ko), moins galère et largement suffisante pour ce qui nous concerne. Bref, en ouvrant le fichier **rc.hlp** vous trouverez le lien Resource-Definition Statements puis **ICON Resource**. Cette rubrique vous permettra de comprendre ce qui est écrit dans notre fichier **testres.rc** et aussi d'aller voir quelles sont les autres ressources à notre disposition. On remarquera aussi que le fichier **b5ms.zip** contient une version plus lourde (23,6 Mo) et plus récente du fichier **win32.hlp** mentionné plus haut sur cette page.

La syntaxe indiquée dans le fichier **rc.hlp** de notre ressource icône pour le fichier **testres.rc** est **nameID ICON [load-mem] filename** où **nameID** peut être soit un nombre entier, soit un nom (chaîne de caractères). Dans notre cas on choisira un nom. Ce nom sera indiqué dans notre fichier principal de gestion de fenêtre **testres.c** au niveau de la déclaration de la classe de fenêtre. Voici la partie du programme qui correspond à la déclaration de la classe de fenêtre :

```
cls.hInstance = hInstance;
cls.lpszMenuName = lpszAppName;
cls.lpszClassName = lpszAppName;
cls.hIcon = LoadIcon(hInstance, "MAINICON");
cls.hCursor = LoadCursor(NULL, IDC_ARROW);
cls.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
cls.style = CS_VREDRAW | CS_HREDRAW;
cls.lpfWndProc = (WNDPROC)WndProc;
cls.cbWndExtra = 0; cls.cbClsExtra = 0;
```

Je vous ai marqué volontairement **MAINICON** en caractères gras pour vous montrer qu'on doit le retrouver à la fois dans le fichier de ressources **testres.rc** et le fichier principal (patati et patata...) **testres.c**. en recherchant la syntaxe de la fonction **LoadIcon()** dans le fichier **win32.hlp** (elle y figure, c'est donc une fonction API point final) on trouve :

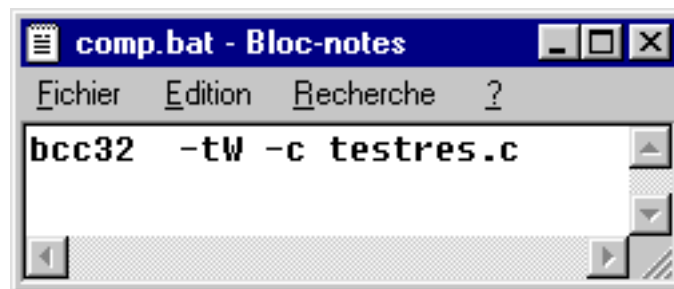
```
HICON LoadIcon( HINSTANCE hInstance, // handle of application instance
                LPCTSTR lpIconName // icon-name string or icon resource identifier
                );
```

Où **lplconName** (remplacé par **MAINICON** dans notre programme) doit être une chaîne de caractères du nom d'une ressource icône. c'est donc gagné ! il ne nous reste plus qu'à compiler **testres.c** avec **bcc32.exe** pour en faire un **testres.obj** c'est à dire un fichier objet (c'est comme ça qu'on doit l'appeler. C'est tout !!!), **testres.rc** avec le compilateur de ressources **brc32.exe** pour en faire un **testres.res** c'est à dire un fichier de ressources compilé, pour ensuite lier **testres.obj** avec **testres.res** en utilisant l'éditeur de liens (on peut dire aussi linker) **ilink32.exe** et ainsi réaliser notre exécutable **testres.exe** muni de sa belle icône (intéressant non ?).

brc32.exe et **ilink32.exe** tout comme **bcc32.exe** se trouvent dans le dossier d'installation de **Borland C++ 5.5** : **c:\Borland\Bcc5\Bin** par conséquent nous allons pouvoir réaliser les **batch** de compilation correspondants et les exécuter un par un depuis l'explorateur parce que nous avons défini le **path** de **c:\Borland\Bcc5\Bin** dans l'**autoexec.bat**. Vous trouverez toute la documentation disponible sur **bcc32.exe**, **brc32.exe**, **ilink32.exe** dans le fichier d'aide **bc55tool.hlp** fourni avec le compilateur gratuit **Borland C++ 5.5**.

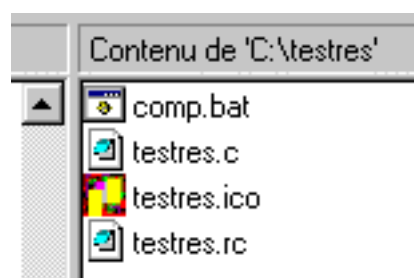
En route pour compiler testres.c

Voici ce que le fichier batch **comp.bat** doit contenir :



Par rapport à la compilation de **fenbase.c** nous avons simplement ajouté l'option **-c** pour empêcher le compilateur de lancer **ilink32.exe** tout seul et l'obliger à ne créer que le fichier **testres.obj**. Bon, dans cet ordre, ça n'aurait pas changé grand chose si on n'avait pas ajouté **-c** parce que le **testres.exe** créé (sans l'icône souhaitée) aurait été écrasé par le suivant (celui de plus tard avec son icône souhaitée). Mais c'est plus propre comme ça.

Par conséquent le contenu de **c:\testres** est le suivant :



Et après un double clic sur **comp.bat**, quelques secondes de patience et un clic sur la croix de la fenêtre DOS pour la fermer lorsque la compilation est terminée on obtient la création du fichier **testres.obj** :



VI-B - En route pour compiler testres.rc

Le compilateur de ressources **win32** est donc **brc32.exe** (on n'en avait pas encore beaucoup parlé de celui-là). Si vous tapez **brc32** sous DOS ou dans une fenêtre Dos, le compilateur de ressources affiche ses options de ligne de commande :

```

C:\WINDOWS>brc32
Borland Resource Compiler / Binder
Version 5.40 Copyright (c) 1992, 1999 Inprise Corporation

Syntax: BRC32 [options ...] filename
options marked with a '*' are on by default

-r                compile only. Do not bind resources
-fofilename       set output res filename
-fefilename       set output exe filename
-v               verbose
-ipath            set include path
-x               ignore INCLUDE environment variable
-dname[=string]  define #define
-32              * build 32-bit Windows compatible res/exe files
-16              build 16-bit Windows compatible res/exe files
-Ud.d Mark the .exe file with Windows version provided (4.0 is the default)
-31 Provided for downward compatibility (build 16-bit res/exe files)
-w32 Provided for downward compatibility (build 32-bit res/exe files)
-k               do not create fastload area (16 bit only)
-t               (ignored, for compatibility)
-? or -h        display this message

C:\WINDOWS>

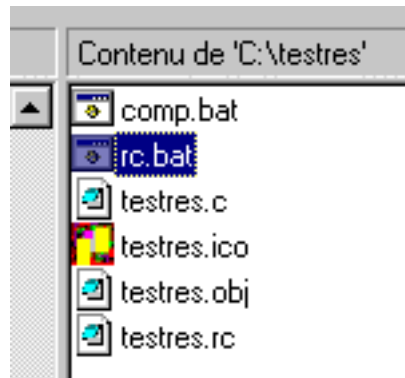
```

Voici ce que le fichier **batch rc.bat** (rc pour resource compilation ça fait plus fun !) pour la compilation de **testres.rc** doit contenir :



Comme indiqué plus haut, l'option **-32** indique une compilation **win32** et l'option **-r** de ne produire que **testres.res**.

Par conséquent le contenu de **c:\testres** est le suivant :



Et après un double clic sur **rc.bat**, quelques secondes de patience et un clic sur la croix de la fenêtre DOS pour la fermer lorsque la compilation est terminée on obtient la création du fichier **testres.res** :



En route pour l'édition de liens (ou linkage)

Rappel : Le but de l'opération ici est de lier **testres.obj** à **testres.res** pour en faire **testres.exe**. Voici ce que le fichier batch **link.bat** (devinez pour link !) doit contenir :

```
ilink32 -aa c0w32 testres.obj,testres,,import32 cw32,,testres.res
```

Alors ça vous en bouche un coin ?

Allez donc plutôt faire un tour dans le fichier **bcb5tool.hlp** fourni avec le compilateur gratuit **Borland C++ 5.5** (je crois que je me répète) pour y voir un peu la syntaxe que doit prendre la ligne de commande de **ilink32**. Voici ce qu'on y trouve :


```
ILINK32 [@respfile][options] startup myobjs, [exe], [mapfile], [libraries], [deffile],  
[resfile]<br/>
```

Eh oui la place des virgules est très importante. il suffit d'en oublier une et patatra...

Ce qui est entre crochets n'est pas indispensable c'est écrit dans la doc !

[@respfile] = fichier de réponse du linker (on verra comment en faire un à notre sauce plus tard) pour l'instant on va s'en passer.

[options] = **-aa** : Construire une application Windows 32 bits (important non ?). Mieux vaut être sur !

startup = c0w32 c'est obligatoire c'est tout ! on trouve ce fichier (c0w32.obj) dans le dossier **c:\Borland\Bcc5\Lib** du compilateur. Il paraît que c'est pour bien initialiser le linker et tout le reste en vue de la création d'un programme Windows dans les normes.

myobjs = Ah voilà où il faut placer **testres.obj** !

[exe] = testres : On va lui dire que notre programme devra s'appeler **testres.exe** au cas où il ne s'en souviendrait pas !

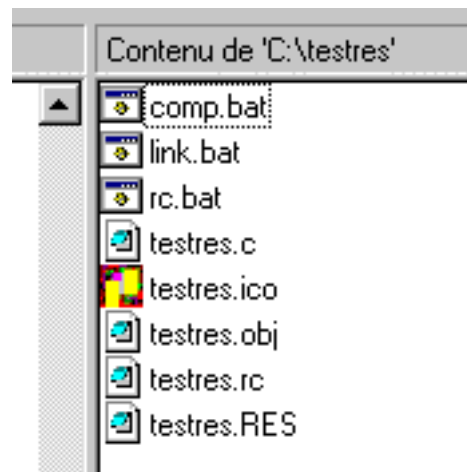
[mapfile] = Moi j'y mets rien. Il pourra faire un fichier de mapping si ça lui fait plaisir et aussi choisir le nom correspondant tant qu'il y est. Moi ça m'est égal.

[libraries] = import32 cw32 : bon, ils disent que ce qui est entre crochets n'est pas indispensable mais si on n'écrit pas ici les références à **import32.lib** et **cw32.lib** le linker n'ira jamais chercher les entrées des fonctions de l'API dans ces fichiers et on se demandera pourquoi **testres.exe** ne marche pas : Ce programme va être arrêté parcequ'il a réalisé une opération non conforme (et j'en passe !).

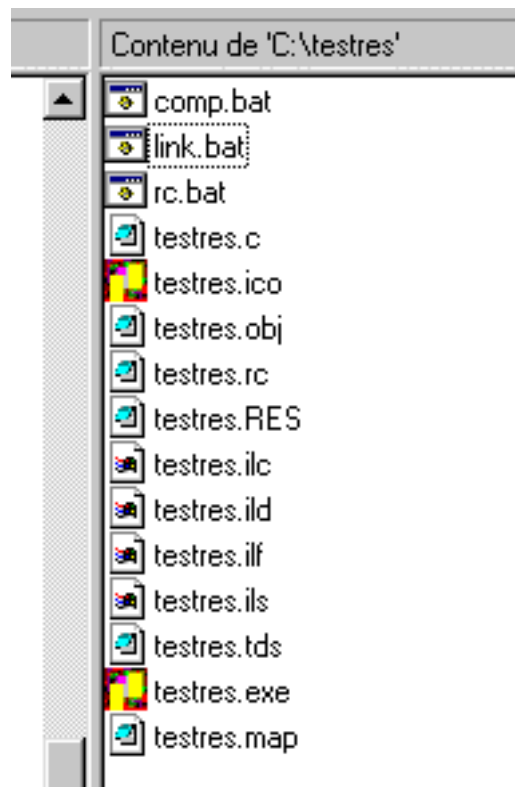
[deffile] = Si on veut se faire plaisir et construire un fichier de définition de module qu'on référencera ici on peut. Moi je ne fais rien. Le linker n'aura qu'à se débrouiller avec ses options par défaut.

[resfile] = Ah voilà où il faut placer **testres.res** !

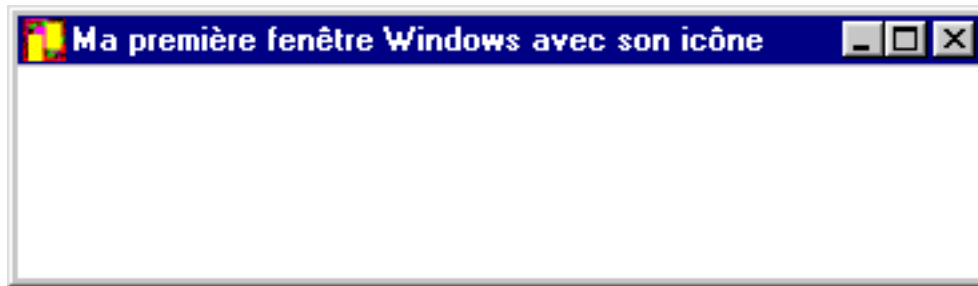
Par conséquent le contenu de **c:\testres** est le suivant :



Et après un double clic sur **link.bat**, quelques secondes de patience et un clic sur la croix de la fenêtre DOS pour la fermer lorsque l'édition de liens (ou linkage) est terminée on obtient la création du fichier **testres.exe** :



Miracle ! voici notre petit **testres.exe** sur lequel on peut double cliquer pour voir s'afficher notre fenêtre :



Bon je l'ai volontairement un peu réduite il faut bien l'avouer. (Là aussi je me répète ?)

Réaliser un fichier batch de projet

Pour développer notre fenêtre Windows avec son icône nous avons décomposé les différentes étapes :

- Compilation du fichier principal de gestion de fenêtre **testres.c** avec **comp.bat**.
- Compilation du fichier de ressources **testres.rc** avec **rc.bat**.
- Edition de lien avec **link.bat**.

Afin de nous éviter toutes ces manipulations (fastidieuses il faut bien le dire), nous allons regrouper toutes les commandes en un seul fichier appelé **projet.bat** (c'est moi qui l'ai inventé). De plus, on fera en sorte que les messages affichés par les compilateurs et le linker pendant leur travail soient placés dans des fichiers qu'il sera possible d'aller lire autant de fois qu'on en aura envie pour la recherche d'éventuelles erreurs (personne n'est parfait).

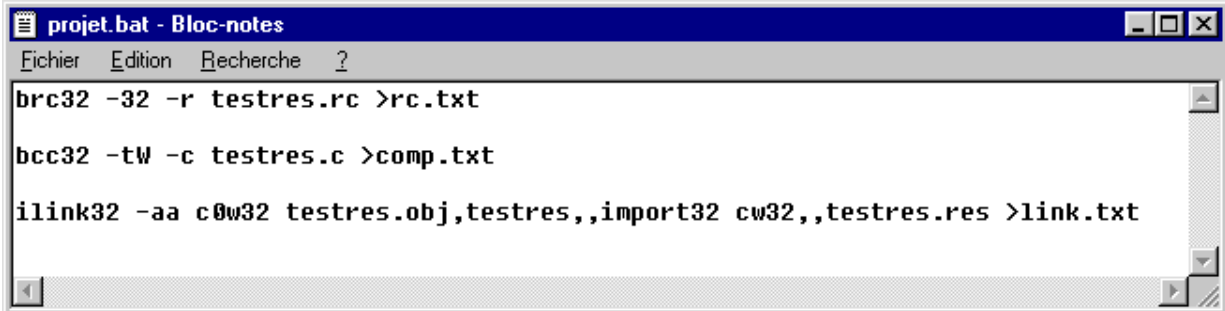
Tant qu'on y est pour la recherche des erreurs, je voulais vous indiquer que le fichier **BCB5ERRS.HLP** contenu dans **b5std.zip** pouvait vous être très utile (Quand vous voudrez programmer tout seul dans votre coin et que vous refuserez qu'on vienne vous donner des conseils).

Pour télécharger **b5std.zip** :

<ftp://ftp.borland.com/pub/borlandcpp/techpubs/bcc55/b5ctool.zip>

<http://www.borland.com/techpubs/borlandcpp/v55/updates/cmd.html>

Voici ce que le fichier batch **projet.bat** doit contenir :

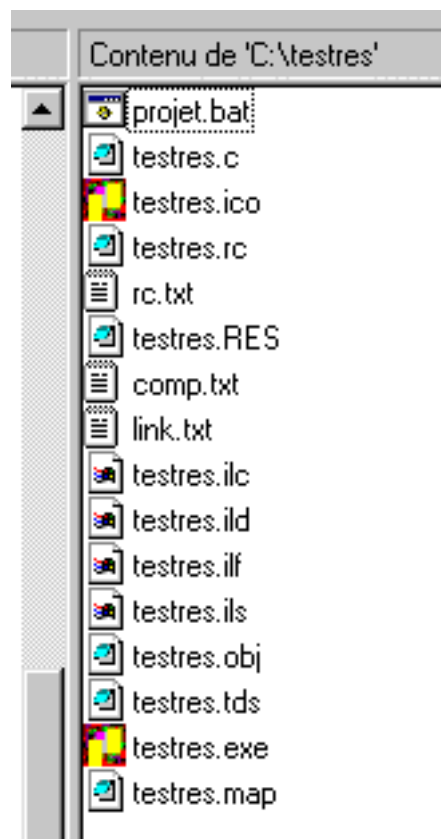


```
projet.bat - Bloc-notes
Fichier  Edition  Recherche  ?
brc32 -32 -r testres.rc >rc.txt
bcc32 -tW -c testres.c >comp.txt
ilink32 -aa c0w32 testres.obj,testres,,import32 cw32,,testres.res >link.txt
```

On se rendra compte que ce n'est ni plus ni moins que le contenu des fichiers batch **comp.bat**, **rc.bat** et placés à la suite. Nous avons simplement ajouté un > de redirection vers un fichier de sortie pour chaque étape. Les informations qui étaient affichées dans la fenêtre DOS lorsque nous procédions de manière indépendante seront enregistrées dans ces fichiers. Si on réinitialise le dossier **c:\testres** avec uniquement les dossiers nécessaires voici ce qu'il doit contenir :



Et après un double clic sur **projet.bat**, quelques secondes de patience et un clic sur la croix de la fenêtre DOS pour la fermer lorsque la compilation est terminée on obtient :



C'est à dire la même chose qu'avant avec en plus les fichiers de sortie **comp.txt**, **rc.txt**, **link.txt**, le tout en une seule opération. Le fichier **testres.exe** est aussi disponible et il ne reste plus qu'à cliquer dessus pour ouvrir notre belle fenêtre, faire un petit raccourci à placer sur le bureau pour mieux voir son icône etc...

Il ne vous reste plus qu'à vous plonger dans les livres et les fichiers d'aide. Le descriptif des fonctions standard du **langage c** se trouve dans le fichier **bcb5rtl.hlp** contenu dans **b5std.zip** vu plus haut. Lorsque vous pensez pouvoir utiliser une fonction vous devez vous assurer qu'elle soit compatible win32 en cliquant sur Portability. N'oubliez pas aussi d'inclure le fichier d'en-tête contenant la fonction au début de votre programme comme nous l'avons fait pour **windows.h**. Vous avez ici tous les éléments dont vous avez besoin pour travailler à réaliser de belles applications sans dépenser un sou. Avec un bon éditeur de code gratuit comme **SynEdit** vous pourrez faire tout ce que vous voulez. Il suffit de lui indiquer le fichier **projet.bat** dans son menu **Advanced -> Invoke DOS command** pour ensuite lancer la compilation en pressant la touche **F7** puis **enter**.

J'ai trouvé **SynEdit** sur <http://pcoudert.developpez.com/bc55tut/http://www.download.com> en utilisant le mot clé **editor** et pour le télécharger il suffit de cliquer sur le lien suivant : <http://synedit.hypermart.net/synedit.zip> **SynEdit** pèse 3.28 Mo.

Vous pouvez aussi vous connecter sur <http://www.webnotes.org/> pour voir ce qu'il est possible de faire en **C** avec **Borland**, trouver des exemples de code <http://www.webnotes.org/sources.htm>, utiliser les moteurs de recherche pour trouver un fichier particulier sur internet <http://www.webnotes.org/chercher.htm> etc...