Tutoriel sur la programmation Batch

Par Adrien REBOISSON - rabusier@aol.com

Table des matières

VI] Boucles

La commande For… Do…

VII] La compilation

Qu'est ce que la compilation
Comment compiler un fichier Batch
Les erreurs de compilation

VIII] Les bordures

Générer des bordures en utilisant l'assistant dans PowerBatch

IX] Ecriture dans des fichiers

Écriture en mode ajout
Écriture en mode écrasement
Écriture de résultats de commande
Redirection vers le périphérique virtuel NUL

X] Appel d'autres fichiers Batch

Utilisation de fichiers Batch come sous-programme
Lancement d'autres fichiers Batch

XI] Travail avec ERRORLEVEL

Utilisation de la commande ERRORLEVEL

XII] 5 autres fonctions de PowerBatch

Test ligne, test bloc, test pas à pas
Les modèles
L'assistant XCOPY
La commande CHOICE
Le convertisseur HTML

Avertissement : Ce tutoriel n'a pas pour vocation de remplacer un livre dédié à la programmation Batch, mais surtout d'initier le programmeur débutant à cette technique. Il n'est pas exempt d'erreurs, si vous en repérez, merci de me contacter par mon e-mail

rabusier@aol.com

La version la plus récente de ce manuel sera toujours publiée sur http://astase.com4.ws

1°) Introduction

Basiquement, un fichier Batch n'est rien de plus qu'un fichier texte contenant des commandes MS-DOS, et possédant le suffixe ".bat".

Si vous ne connaissez pas MS-DOS ou n'avez jamais entendu parler de *Autoexec.bat,* passez votre chemin : en effet, la programmation Batch nécessite une connaissance minimum de l'environnement DOS.

En fait, un fichier Batch contient simplement une suite de commandes que vous pourriez taper sous l'invité (prompt) du DOS, chaque nouvelle ligne du fichier correspondant à une nouvelle commande. Néanmoins, certaines commandes ne sont qu'utilisables dans les fichiers batch du fait de leur inutilité dans l'environnement de commande DOS.

Leur utilité est, par exemple, quand il faut répéter toujours la même série de commandes. À titre d'exemple, nous pourrions évoquer le changement de répertoire et peut-être aussi la commande FORMAT qu'on fait souvent suivre de la commande CHKDSK pour vérifier si la disquette a bien été formatée.

Exemple :

Imaginons un fichier batch contenant les commandes suivantes :

```
cd \
cd games
superjeu.exe
```

Cela aurait le même effet que si vous tapiez sous DOS les commandes suivantes :

```
C:\Chemin> cd \ [Entrée]

C:\> cd games [Entrée]

C:\games> superjeu.exe [Entrée]
```

L'intérêt des batch est donc d'automatiser des tâches répétitives effectuées sous DOS.

Les fichiers batch sont donc très faciles à créer puisqu'un simple éditeur texte suffit (Comme EDIT, sous DOS)
Les fichiers batch peuvent également utiliser toutes les commandes DOS, ce qui rend disponible pour le programmeur un grand nombre de fonctions.
Enfin, leur taille est relativement légère par rapport à d'autres programmes, ce qui facilite leur "transferts" sur différents disques et supports de stockage.

Cependant…

- Le langage Batch n'est pas compilé, il est interprété par COMMAND.COM ce qui rend plus lent l'exécution de programmes batch par rapport à des applications écrites directement en langage machine,

- Les fichiers Batch sont directement éditables, donc votre code n'est pas "protégé" à la copie par d'autres programmeurs,

- Enfin, et surtout, des opérations élémentaires telles que le traitement de chaînes de caractères, d'opérations mathématiques, etc… n'existent pas sous DOS, ce qui implique l'usage de programmes externes (s'ils existent, selon les cas).

2°) Création de fichiers Batch avec PowerBatch

Il existe un logiciel nommé *PowerBatch* permettant de créer très facilement des fichiers Batch, en utilisation libre en plus.

Nous allons apprendre à nous en servir pour créer nos fichiers Batch.

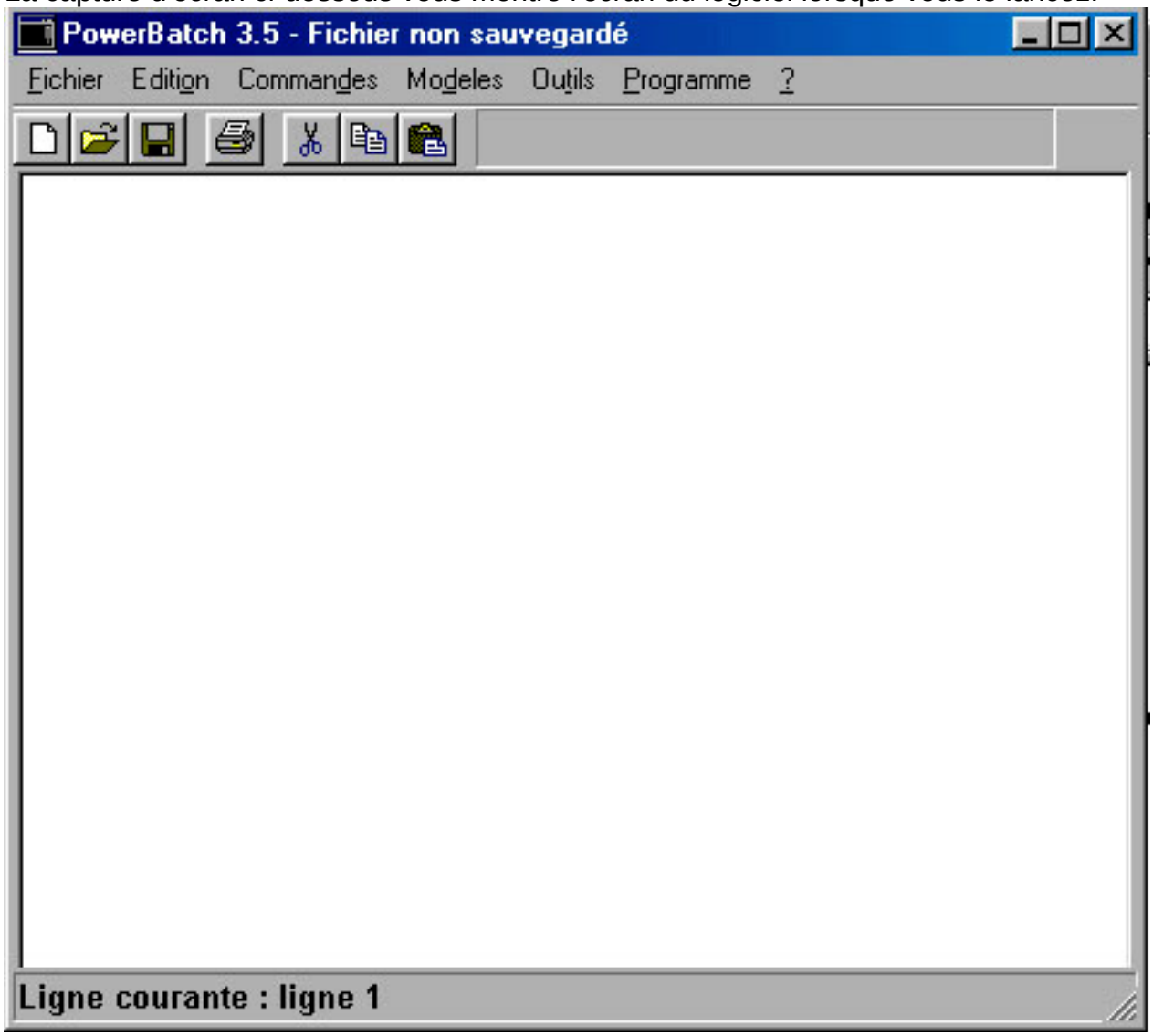
1°) Télécharger le fichier depuis http://astase.com4.ws

2°) Décompressez le fichier ZIP, lisez attentivement les fichiers "Lisez – moi", puis installez le logiciel (Si besoin est, vous devez à un moment redémarrer votre PC puis recommencer l'installation).

1°) Création du fichier Batch "Hello, Word"

Nous allons nous atteler à la programmation d'un fichier affichant à l'écran le traditionnel "Hello Word" en langage Batch.

La capture d'écran ci-dessous vous montre l'écran du logiciel lorsque vous le lancez.



Dans la zone de texte, tapez :

```
Echo Hello Word !
```

Qu'est ce que vous venez d'écrire ?

• Vous avez écrit la commande "Echo" permettant d'afficher du texte à l'écran.
Cette commande exige un paramètre : le texte qu'elle doit afficher à l'écran.
Le paramètre est donc placé à droite de la commande, séparé par un espace.

En réalité, ECHO est utilisé pour faire sortir tous types de données dans n'importe quel périphérique (et même dans un fichier). Dans notre cas nous l'utilisons pour faire sortir des données sur l'écran d'un ordinateur, nous allons donc dire pour l'instant que ECHO est une commande permettant d'afficher du texte à l'écran.

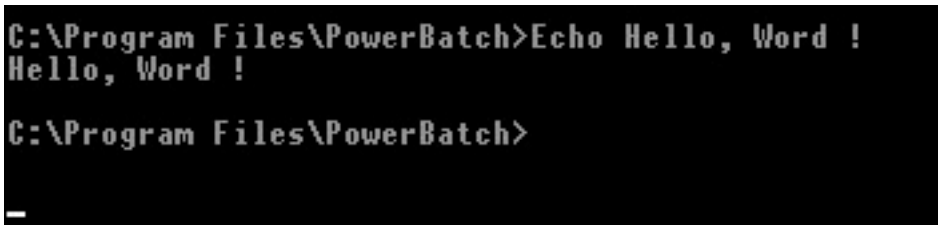
Remarquez l'absence de guillemets, par rapport à d'autres de langages de programmation exigeant que les variables littérales soient distinguées par des guillemets.

NOTE IMPORTANTE : MS-DOS n'est pas sensible pour les commandes à la différence entre les majuscules et les minuscules, que vous écriviez echo ou Echo, ou bien encore ECHO ou EcHo, le résultat sera le même.

PowerBatch possède une fonction bien pratique permettant de tester le fichier Batch en cours.

Pressez F6, ou exécutez la commande "Test global" situé dans le sous-menu "Tests" du menu "Programme".

Une fenêtre DOS apparaît, avec le resultat suivant :



Le DOS nous a bien affiché notre résultat, MAIS il apparaît comme si on venait d'entrer les commandes séparément sous DOS : En effet, on distingue l'invite (*c:\Program Files\PowerBatch>)*, la commande (*Echo Hello, Word !),* son résultat dessous, puis un second invite.

Nous souhaiterions que seuls les résultats des commandes apparaissent à l'écran.

Il va falloir utiliser *l'écho local*. L'écho local est une fonction permettant ou non de voir uniquement les résultats des commandes entrées.

Ci-dessus, l'écho local est activé, puisque l'on voit l'invite DOS et les commandes comme si on les avait tapés sous DOS.

Il va falloir désactiver cet écho en tapant :

```
Echo off
```

Qu'est ce que vous venez d'écrire ?

• Vous avez écrit la commande "Echo" permettant d'afficher du texte à l'écran, mais vous avez transmis un paramètre particulier à la commande : il s'agit du paramètre "off", qui désactive l'écho local. Cette commande accepte aussi le paramètre "on" qui permet d'activer cet écho. Vous avez donc dans le cas présent désactivé l'écho local.

Testez de nouveau le fichier en écrivant donc :

```
Echo off
Echo Hello Word !
```

Puis pressez ensuite F6 :

Voilà comment le programme s'est déroulé :

- L'écho est sur ON : le programme affiche toutes les commandes avant de les exécuter. Là, le programme a rencontré la commande "echo off". Il l'a affiché, puis l'a executé. L'écho est maintenant sur OFF, il est désactivé.
- Le programme rencontre la commande "Echo Hello, Word !". Il se contente donc d'afficher "Hello, Word !" à l'écran.

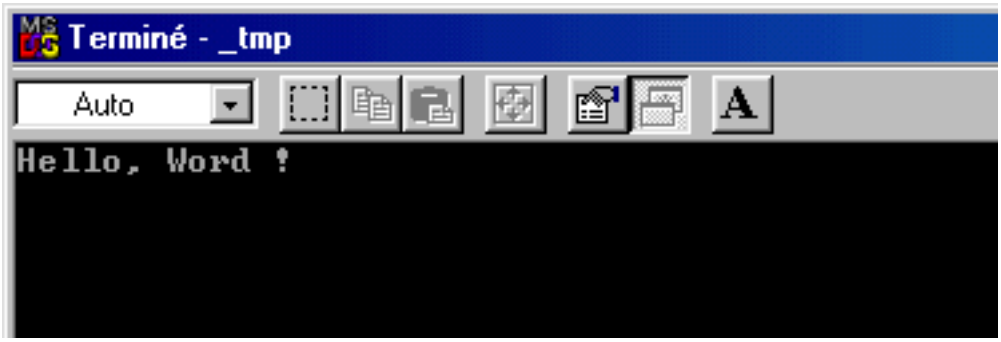
Néanmoins, on voit toujours l'invite en haut, ainsi que "echo off". Il nous faut donc trouver un moyen de les supprimer.

La commande "@" est adapté à notre cas : elle permet de désactiver immédiatement l'écho pour une ligne, il suffit juste de faire précéder la ligne de ce signe.

On a donc :

```
@Echo off
Echo Hello Word !
```

Pressez F6 pour executer le fichier et batch… Et là, on a enfin que ce que l'on cherche :



A retenir…

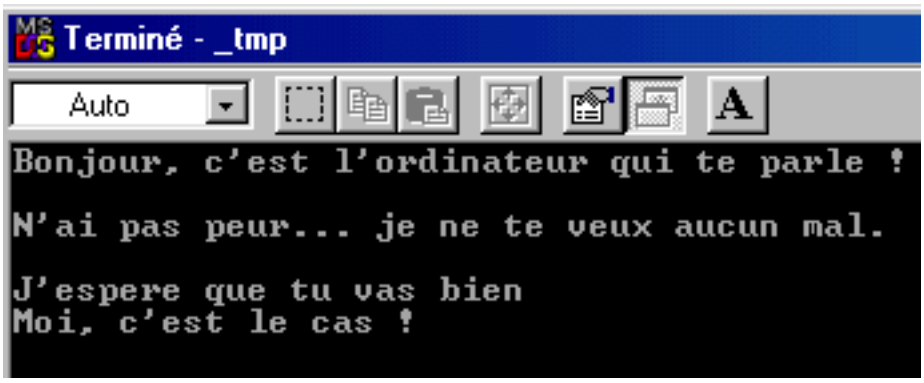
- La commande "echo", pour afficher un texte,
- Qu'est ce que l'écho local,
- Le paramètre "off" pour désactiver l'écho local, "on" pour l'activer,
- Le caractère "@" pour désactiver l'écho local sur une ligne.

Aller plus loin :

La commande "echo." permet d'afficher une ligne vide.
On peut donc afficher un petit texte, par exemple :

```
@Echo off
Echo Bonjour, c'est l'ordinateur qui te parle !
Echo.
Echo N'ai pas peur… je ne te veux aucun mal.
Echo.
Echo J'espere que tu vas bien.
Echo Moi, c'est le cas !
Echo.
```

Donne :



Commentez votre code

Essayons de prendre les bonnes habitudes tout de suite…
Pour vous et sur le moment, votre code vous paraît parfaitement clair. Mais le sera t-il dans un mois ou lu par une autre personne ?

Il est donc nécessaire d'insérer des remarques, appelées "commentaires" dans vos fichiers.

La commande "REM" (REMark) est là pour vous ! Il suffit simplement d'apposer votre commentaire après la commande, par exemple :

```
@echo off
REM  Formatage de la disquette
Format a:
REM  Creation du dossier Backup sur a:\
Mkdir a:\backup\
REM  Copie des fichiers
Copy c:\backup\*.* a:\backup\*.*
```

La commande REM ne sera pas exécutée, mais vous aidera à mieux comprendre ce que vous avez voulu faire lorsque vous n'avez pas étudié le Batch depuis longtemps.

Attention aux caractères spéciaux !

Les caractères spéciaux sont les accents, signes divers comme /, %, etc.

N'utilisez pas d'accent, car MS-DOS va remplacer les caractères accentués par des symboles :

```
@echo off
echo J'ai été reçue à mon examen !
```

Donne :

```
J'ai útú reÞue ó mon examen !!!
```

Utilisez plutôt la fonction d'accentuation de PowerBatch (Menu "Commandes", "sous-menu "Caractères spéciaux", articles "Accent grave", "Accent circonflexe", "Accent aigu", etc.)

A retenir…

- La commande "echo." Pour afficher une ligne vide
- La nécessité de commenter son code (Commande REM)
- Eviter d'utiliser directement des caractères spéciaux, comme des caractères accentués.

Astuce PowerBatch : Utilisez l'article "Standard Batch" du menu "Modèle" pour créer un batch automatiquement avec @echo off.

2°) Utilisations de commandes standard DOS dans un fichier Batch

Il est très facile d'exécuter des commandes DOS dans un fichier Batch, et c'est même fait pour ça…

Il vous suffit d'insérer la commande dans le batch.
Par exemple, voici un fichier listant le contenu du disque C:, allant dans le dossier Jeux, puis exécutant le fichier SuperJeu.exe :

```
@echo off
echo Listage du disque C:\
dir c:
echo Va dans le dossier jeux
cd jeux
echo Lance SuperJeux.exe
Superjeux.exe
```

Notre but n'est pas de vous apprendre toutes les commandes MS-DOS, vous êtes censé connaître les plus communes.

Nous allons donc étudier ce qu'*est* une commande MS-DOS en réalité.

RAPPEL : Nous sommes sous DOS, les noms de fichiers sont limités à 8 caractères (sinon, on tronque les deux derniers caractères par ~x, x représente un nombre "discriminant" destiné à distinguer deux éventuels noms communs)

Il y a deux possibilités :

- Soit la commande DOS est intégrée à l'interpréteur COMMAND.COM. C'est le cas des commandes les plus communes comme DIR, CD, etc…

- Soit la commande est en réalité un exécutable DOS, c'est-à-dire que c'est une application qui est intégrée sous la forme d'une commande standard. C'est sur ce cas que nous allons nous pencher.

1°) Comment un exécutable peut-être considéré comme un commande ?

Une commande est par définition un "mot" que l'on peut entrer où que l'on soit (que l'on soit dans le répertoire A ou le répertoire B), et qui ne nécessite pas qu'on indique son chemin d'accès, et qui bien sûr agit physiquement ou logiciellement sur votre ordinateur directement ou à l'aide de paramètres.

> **RAPPEL :** Pour lancer un fichier .EXE, .COM ou .BAT, il n'est pas nécessaire de préciser l'extension de ces derniers.
> Pour lancer Superjeu.exe, vous n'êtes pas obligé de taper :
>
> ```
> Superjeux.exe
> ```
>
> Vous pouvez simplement entrer :
>
> ```
> Superjeu
> ```

… pour que MS-DOS "comprenne" que vous souhaitez lancer le programme "Superjeu.exe".

Mais ou sont donc stockés ces commandes ?

Ces commandes sont stockées "naturellement" dans C:**DOSSIER_DE_WINDOWS**\COMMAND\, donc, dans la majorité des cas, dans C:**WINDOWS**\COMMAND\

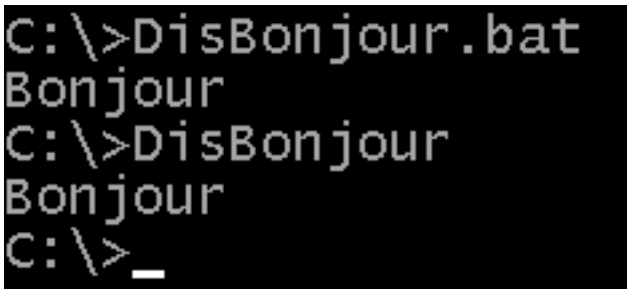
Si vous possédez un fichier .exe, .com ou .bat, et que vous souhaitiez l'établir en tant que commande DOS, copiez – le simplement dans ce répertoire.

Par exemple, prenons l'exemple de DisBonjour.bat

Il contient une commande permettant d'afficher à l'écran "Bonjour".

Copiez-ce fichier dans C:\WINDOWS\COMMAND\

Tapez ensuite **DisBonjour.bat** => Votre texte apparaît à l'écran…
Plus fort : tapez simplement **DisBonjour =>** Votre texte apparaît aussi à l'écran…



Conclusion 1 : Pour "ajouter" des commandes à MS-DOS, copiez des exécutables DOS d'extension .bat, .exe, ou .com dans le répertoire "COMMAND" du dossier de Windows.

2°) La variable PATH et les autres répertoires d'ajout possibles

D'autres répertoires peuvent définir des chemins d'accès potentiels à des commandes.

Pour voir les chemins d'accès possibles installés sur votre machine, tapez "path" dans une session MS-DOS.

Voilà un exemple possible de résultat :

On peut voir que les répertoires d'accès sont au nombre de 3, et séparés par des point-virgules :

C:\Windows (2 fois, il s'agit sans doute d'une erreur d'un logiciel)
C:\Windows\Command
C:\Cntx

Cela veut dire que n'importe quel fichier .exe, .bat, ou .com peut être lancé comme une commande dans l'environnement DOS :

Conclusion 2 (et finale) : Pour "ajouter" des commandes à MS-DOS, copiez des exécutables DOS d'extension .bat, .exe, ou .com dans un des répertoires spécifiés par la variable "Path".

3°) Ajouter un chemin d'accès à la variable path

"Path" est une variable d'environnement, c'est à dire, (nous le verrons plus en détails plus tard) que cette variable représente une valeur accessible n'importe ou et n'importe quand dans l'environnement DOS.
Dans le cas actuel, cette variable est modifiable, nous allons donc inclure un autre chemin d'accès dans cette variable.

REGLE 1 : Une variable d'environnement est identifiée à la lecture lors d'une définition par deux "%" autour de lui. En effet, à l'execution, MS-DOS remplace le contenu d'un "mot " entouré par deux "%" par sa valeur.

Dans le cas actuel, "Path" peut-être lue en appellant "%PATH%".

Pour ajouter un chemin d'accès, on fera donc :

PATH=%PATH%;*CHEMIN_A_AJOUTER*

En effet :

- Le premier PATH est en écriture (pas besoin de %PATH%)
- Le second PATH est destiné à inclure l'ancien path, et doit contenir la variable PATH (d'où %PATH%)
- Le point-virgule est destiné à séparer le chemin précédent du nouveau chemin.
- CHEMIN_A_AJOUTER représente le chemin d'accès à ajouter

Imaginons que la variable PATH contienne "C:\WINDOWS;C:\WINDOWS\COMMAND". Nous souhaitons ajouter le chemin C:\MESJEUX\SUPERJEUX

On inscrira donc dans un fichier Batch ou directement dans l'interprêteur COMMAND.COM :

PATH=%PATH%;C:\MESJEUX\SUPERJEUX

Ce qui donne pour le DOS :

PATH=***C:\WINDOWS;C:\WINDOWS\COMMAND***;C:\MESJEUX\SUPERJEUX

Maintenant, imaginons que vous souhaitez lancer Superjeu.exe situé dans C:\MESJEUX\SUPERJEUX. Or, on vient de mettre le chemin dans le Path.

Par conséquent, on peu simplement taper :

```
Superjeu.exe
```

Ou, comme une commande standard :

```
Superjeu
```

Cela signifie aussi que tous les autres fichiers situés dans le "%path%" pourront-être lancés comme des commandes standard.

Par exemple, si "c:\WINDOWS" est dans le Path, entrez Winver pour lancer C:\windows\winver.exe et afficher la version de Windows (Bien sur ce programme n'est pas fait pour le dos, c'est donc Windows qui le lancera automatiquement).

4°) Paramètres envoyées à une commande ou à un programme.

On appelle paramètre tous les arguments passés à un programme ou une commande. Les paramètres sont séparés par des espaces.

Par exemple, dans :

FORMAT a: /V[:MaDisquette] /B /C

FORMAT est la commande,
A: est le premier paramètre
/V[:MaDisquette] est le second paramètre
/B est le troisième paramètre
/C est le quatrième paramètre.

En fait, FORMAT est un programme (FORMAT.EXE) localisé dans C:\WINDOWS\COMMAND

Ce programme reçoit donc comme argument tous les paramètres envoyés par l'intermédiaire du DOS.

Votre Batch peut, on le verra plus en détail plus tard, recevoir neuf paramètres séparés par des espaces, comme un programme standard comme FORMAT.EXE peut le faire. Ce sont des variables d'environnement spéciales destinées spécifiquement au fichier Batch qui est utilisé : chaque Batch peut donc lire les arguments, si il y en a, qui lui sont envoyés lors de son lancement.

À RETENIR : Dans Windows, il est impossible d'envoyer des paramètres en double cliquant sur un fichier.
Pour envoyer des paramètres à un fichier dans Windows, vous devez le faire soit par un raccourci, en éditant la "destination" du lien, soit par un fichier PIF (*.pif) permettant se définir les préférences d'exécution d'une application DOS.

Vous pouvez donc lire 10 variables relatifs au arguments passés à votre programme.

Ces variables vont de *%0* à *%9* . La variable %0 contient le chemin d'accès au programme, %1 le premier paramètre, %2 le second paramètre… jusqu'à %9 qui contient le neuvième paramètre envoyé au batch.

Exemple : créez avec PowerBatch un fichier ressemblant à celui-ci dessous, puis utilisez la fonction "Test avec paramètres" de PowerBatch (Menu "Programme", sous-menu "Tests") pour envoyer des paramètres au fichier (ou procédez par l'intermédiaire du DOS) :

```
@echo off
echo L'adresse de ce fichier est %0
echo Le premier parametre est %1
echo Le second parametre est %2
echo Le troisieme parametre est %3
echo Le quatrieme parametre est %4
```

Dans le cas ou vous n'envoyez aucun paramètre (vous lancez simplement le fichier), vous obtenez un résultat de ce type :

```
L'adresse de ce fichier est C:\PROGRA~1\POWERB~1\RESSOU~1\TMP\_TMP.BAT
Le premier parametre est
Le second parametre est
Le troisieme parametre est
Le quatrieme parametre est
```

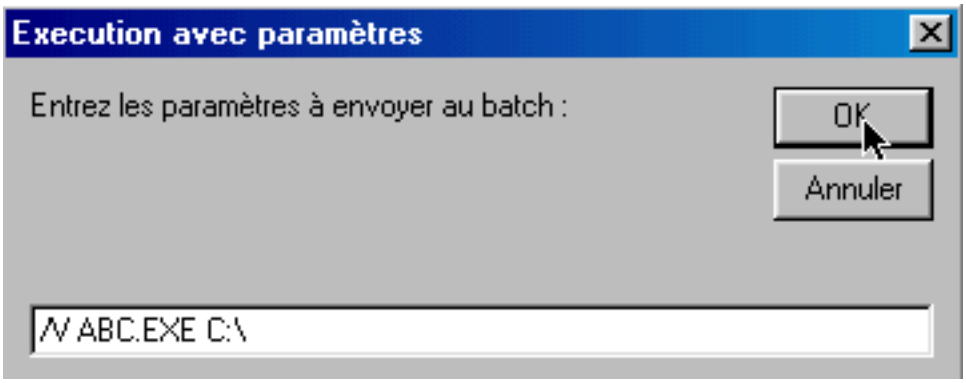
Comme vous le constatez, rien n'apparaît à la place des %1 %2 %3 et %4 : en effet, nous n'avons pas envoyé de paramètre à l'application, c'est donc normal.

Envoyez 3 paramètres, par exemple "/V" pour le premier paramètre, "ABC.EXE" pour le second, et "C:\" pour le troisième.

Sous DOS, vous pouvez lancer le fichier en le faisant précéder de son adresse, puis en envoyant les paramètres, par ex :

```
C:\Tests\Monbatch.bat /V ABC.EXE C:\
```

Vous pouvez procéder plus facilement avec PowerBatch : entrez simplement ces paramètres dans la boite de dialogue affichée lorsque vous cliquez sur l'article "Global avec paramètres" dans le sous-menu "Test" du menu "Programme", cliquez sur "OK": le fichier est exécuté avec les paramètres entrés.



Ce qui donne :

```
L'adresse de ce fichier est _tmp.bat
Le premier parametre est /V
Le second parametre est ABC.EXE
Le troisieme parametre est C:\
Le quatrieme parametre est
```

Ce qui est bien sûr parfaitement logique.

Nous apprendrons plus tard à nous en servir dans un programme : à tester si le fichier a des paramètres, à agir en fonction, etc…

A retenir…

Même si ces notions peuvent vous sembler un peu disparates, elles sont importantes pour aborder la suite de la formation :

- Une commande peut-être un fichier, dont le répertoire est inscrit dans la variable %PATH%
- Un fichier peut-être lancé sans préciser son chemin d'accès si son répertoire est dans la variable %PATH%
- La variable %PATH% est modifiable par le DOS ou un fichier Batch
- Comment un fichier peut recevoir des arguments, et comment y accéder via les variables spéciales %*x*

3°) Variables d'environnement

Une variable d'environnement, nous l'avons déjà dit plus haut, représente une valeur accessible n'importe ou et n'importe quand dans l'environnement DOS.

Pour visualiser les variables d'environnement actives sur votre ordinateur, il vous suffit de taper la commande `set` ce qui donne par exemple :

```
TMP=c:\windows\TEMP
TEMP=C:\windows\TEMP
PROMPT=$p$g
winbootdir=C:\WINDOWS
COMSPEC=C:\WINDOWS\COMMAND.COM
PATH=C:\WINDOWS;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\CNTX;
windir=C:\WINDOWS
BLASTER=A220 I5 D1 H5 T6
```

Précisons que, dans le langage Batch, la seule facon de stocker des données est de les associer à des variables d'environnement. Il n'existe pas de variables "locales" que d'autres fichiers Batch ne pourraient pas connaître (sauf les variables sous la forme %*x*)

Nous voyons donc que 8 variables d'environnement sont définies sur cet ordinateur : *TMP*, *TEMP*, *PROMPT*, *WINBOOTDIR*, *COMSPEC*, *PATH*, *WINDIR*; et *BLASTER*.

Sur ces 8 variables, 7 sont définis par WINDOWS : *TMP* (Répertoire temporaire), *TEMP* (Répertoire Temporaire), *PROMPT* (Invite du DOS), *WINBOOTDIR* (Dossier de démarrage de Windows), *COMSPEC* (Adresse de l'interpréteur de commandes), *PATH* et *WINDIR* (Dossier de Windows).

Il est important de savoir que le contenu de ces variables est détruit une fois l'ordinateur éteint ou la session DOS terminée. Il faut donc, si ces variables doivent-être présentes à chaque session, les définir dans Autoexec.bat (qui est lui lancé à chaque démarrage).

Par exemple, la variable "BLASTER" est définie dans Autoexec.bat

Définir une variable d'environnement

Pour définir une variable d'environnement, faites :

Set *NomVariable* = *Valeur de la variable*

Nous souhaitons définir une variable "VersionWindows" contenant la version de Windows (Dans notre cas Windows 98 SE – (Seconde Edition))

Nous allons donc taper dans le DOS, ou écrire dans un fichier batch :

```
Set VersionWindows = 98 SE
```

Validez la commande puis exécutez le Batch.
Il semble que rien ne se passe : normal, cette commande ne produit pas de résultat visible à l'écran.

Pour voir si notre ajout a été pris en compte, il suffit de taper "set" pour voir si notre variable a été ajoutée à la liste de celles déjà définies sur notre ordinateur.

Dans notre cas, il apparaît :

Notre variable a bien été ajoutée.

Notons que :

- Cette variable ne sera détruite qu'à l'extinction de l'ordinateur ou à la fin de la session DOS
- N'importe quel autre programme peut lire, modifier ou réécrire sur cette variable.

Redéfinir une variable d'environnement

Il suffit de réécrire la commande avec une nouvelle valeur, qui viendra écraser l'ancienne, par exemple :

```
Set VersionWindows = Windows Millenium
```

Dans ce cas, l'ancienne valeur écrasera la nouvelle.

Supprimer une variable d'environnement

Il faut simplement assigner une valeur nulle à la variable, par exemple :

```
Set VersionWindows =
```

La variable est maintenant supprimée.

Utiliser une variable d'environnement

Pour utiliser une variable d'environnement, il faut l'encadrer par des "%".
Il n'y a que la commande SET qui ne demande pas de signe "%" pour l'argument représentant le nom de la variable à définir : en effet, on ne lit pas la valeur de la variable, mais on la définit.

Lors de l'exécution du Batch, lorsque l'interpréteur rencontre un nom encadré de deux "%", il substitue ce nom par la valeur de la variable qu'il représente, si elle existe.

Exemple :

```
Echo %VersionWindows%
Echo La version de Windows est %VersionWindows%
Set VersionWindows = %VersionWindows% - 32 Bits
```

La première ligne va simplement afficher la valeur de la variable "VersionWindow"
La seconde ligne va afficher "La version de Windows est "suivie de la valeur de la variable "Version Windows".
Enfin, la dernière ligne va redéfinir la variable "VersionWindows" par sa valeur, à laquelle on ajoute " – 32 Bits "(Dans notre cas la version devient "98 SE – 32 Bits".

A retenir…

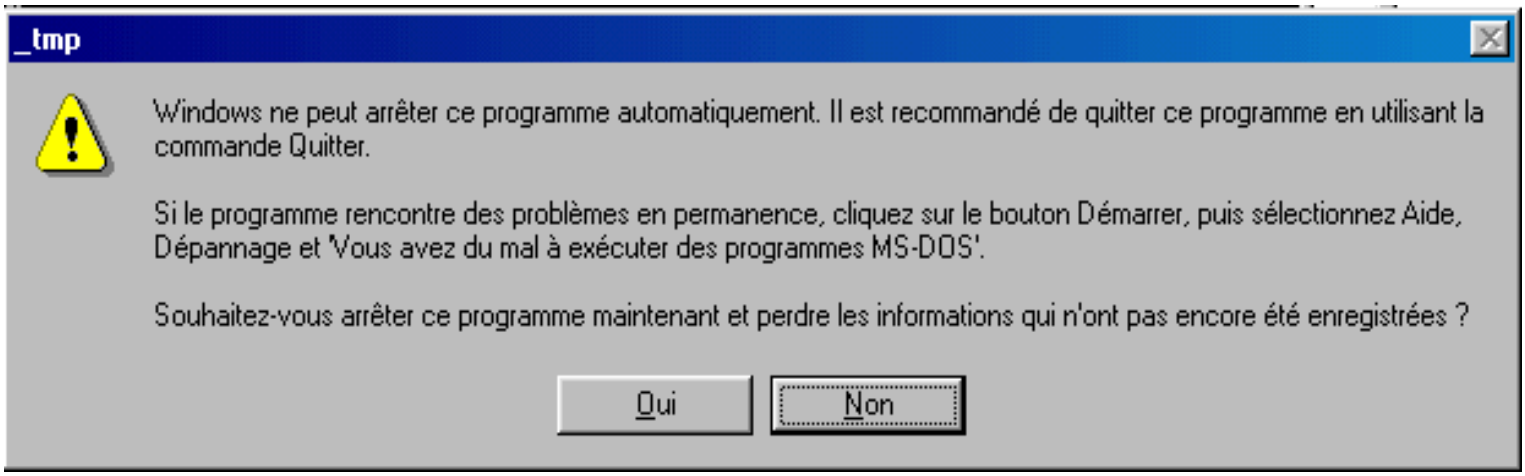
- Qu'est-ce qu'une variable d'environnement
- Comment définir, modifier ou supprimer une variable d'environnement
- Encadrer une variable de "%" pour lire sa valeur
- Les variables d'environnement se perdent à l'extinction de Windows ou à la fermeture de la session DOS. Les variables qui doivent donc être définies à chaque démarrage devront donc être définies dans *Autoexec.bat*

Dans PowerBatch, vous pouvez facilement agir sur les variables en utilisant le sous – menu "Variables" du menu "Commandes"

4°) Saut inconditionnel

Le langage Batch vous permet d'utiliser des commandes de boucle, c'est a dire de répeter un bloc de commandes indéfiniment.

Nous allons étudier dans ce chapitre la commande "Goto". C'est une commande de saut inconditionnelle, qui ne peut être arrêtée (ou à l'aide de commandes que vous ne connaissez pas encore), par conséquent vous allez être amené à fermer de façon "brutale" des programmes DOS, et vous rencontrerez sans doute ce message :



Cela signifie que vous tentez d'arrêter un programme DOS qui est toujours actif. Cliquez sur "Oui" pour quitter le programme.

En principe, les lignes de commande sont traitées les unes après les autres dans un fichier Batch. Toutefois, dans certains cas, on est obligé de sauter des lignes pour reprendre le traitement à un autre endroit du fichier. C'est dans ces cas là que nous allons utiliser les commandes de boucle.

On associe souvent une commande de saut à une commande d'instruction conditionnelle (voir chapitre suivant), ou lorsqu'un bloc de commande doit être répété indéfiniment. C'est sur ce cas que nous allons nous pencher pour l'instant.

Notre première boucle

Pour faire une boucle, il nous faut deux commandes :

- La première est un "Label", c'est-à-dire une étiquette posée dans le programme à l'endroit ou la boucle doit commencer.

- La seconde est la commande Goto, (de l'anglais Go To… qui signifie "aller à") qui, accompagnée du nom du Label, indique à l'ordinateur, quand il doit se rendre à l'étiquette du même nom.

Par exemple :

Commande 1
Commande 2
Label BONJOUR
Commande 3
Commande 4
Commande 5
Goto BONJOUR

Les commandes 1, et 2, sont exécutées une fois, alors que les autres commandes sont exécutées en boucle, puisque le programme rencontre "GOTO", va au label du même nom, continue, rencontre à nouveau "Goto", reva au label , etc…

• Un label se présente sous la forme :

:NomDuLabel

Le nom ne doit pas dépasser 8 lettres (si le nom du label dépasse 8 lettres, seules les 8 premières lettres seront prises en compte), et ne pas être composé d'espaces. Par exemple

:Debut

…est un bon nom pour un label

• Un "Goto" se présente sous la forme de cette commande suivie du nom du label, par exemple :

Goto Debut

… pour aller au label "Début".

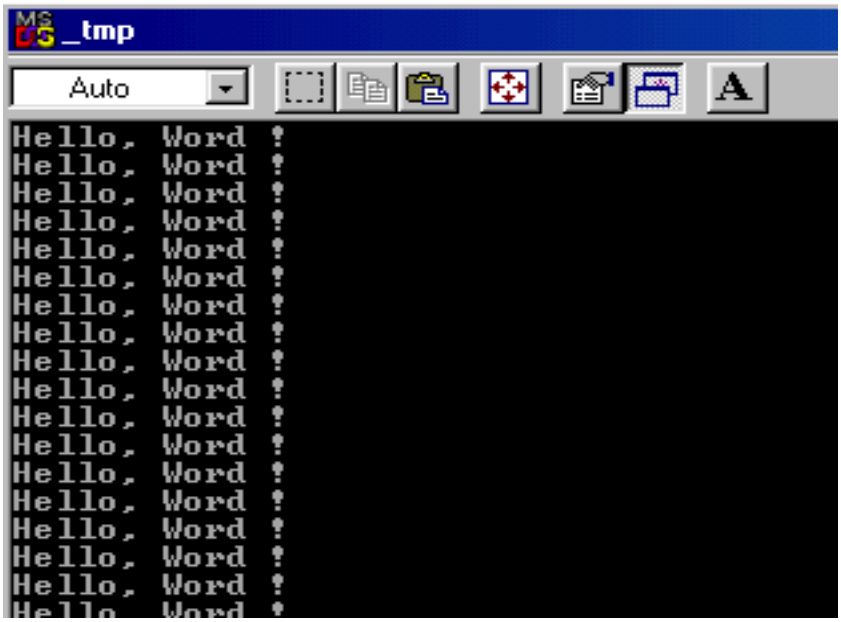
Allons – y pour une boucle infinie !
Nous voulons afficher "Hello, Word !" en boucle.

Nous écrirons donc :

```
@echo off
:Start
echo Hello, Word !
Goto Start
```

Le nom du label est librement configurable, vous pouvez prendre un tout autre nom que "Start", l'essentiel étant que le nom du label et le nom qui suit le "Goto" soient identiques.

Vous obtiendrez un résultat de ce type :



… Signe que notre programme a bien bouclé

A retenir…

- La commande "Goto" permet d'aller au label du même nom
- La marque ":Label" est un "mot" précédé de deux points (":"), correspondant à un repère utilisé par la commande Goto
- Un saut inconditionnel "bouclé "n'est pas "cassable" autrement que par CTRL+C
- Un saut inconditionnel peut aussi être utilisé pour sauter des morceaux de code
- Un saut inconditionnel peut-être utilisé avec la commande "IF" pour exécuter ou ne pas exécuter du code en fonction d'une condition.

5°) Exécution conditionnelle – la commande "IF"

Voici une commande qui permet d'introduire des conditions dans les fichiers batch.

Si la condition formulée est remplie, le reste de la ligne de commande est exécutée, et le programme continue normalement, sinon le reste de la ligne n'est pas exécuté, et le programme continue également.

Attention : seul la fin de la ligne est exécutée, par conséquent seule 1 seule commande peut-être conditionnelle, ce qui peut parfois poser des problèmes. Dans ce cas, utilisez la commande GOTO pour aller à un endroit particulier si la condition est remplie.

Syntaxe d'utilisation :

```
If "<condition>"=="<valeur>" <action>
```

Attention il est important de :

- Toujours encadrer la condition et la valeur à tester par des guillemets,
- De veiller à utiliser, lors d'un test, le **double signe égal** (== au lieu de =)
- Se rappeler que *"<action>"* représente **une seule** commande à executer.

Vous pouvez bien sur comparer des variables avec des valeurs ou comparer des variables ensembles, mais **n'oubliez pas de les encadrer par des guillemets.**

Pourquoi ? Parce qu'à l'exécution, la valeur des variables vient remplacer leur écriture, et si une variable est nulle, MS-DOS génère une erreur car il ne peut comparer un terme qui n'existe pas. Par compte, s'il y a des guillemets, MS-DOS "comprend" qu'il fait une comparaison avec une variable vide.

Exemple :

```
If "%1"=="/AIDE" ECHO Ce texte sera affiche
```

Ici, on va être amené à comparer le contenu de la variable d'environnement paramètre n°1 avec le texte "/AIDE". Si ceux ci sont identiques, un texte sera affiché à l'écran.

Attention à la différence majuscules/minuscule. Même si nous avons dit plus haut que MS-DOS ne faisait pas la différence entre les commandes écrites en majuscules et celles écrites en minuscules, il différencie tout de même les contenus des variables à comparer. Par exemple, si l'utilisateur à entré "/Aide" ou "/aide" au lieu de "/AIDE", la condition ne sera pas validée.

Vous pouvez associer d'autres conditions à la commande IF. Voici les possibilités dont vous disposez :

IF NOT *Condition*

Vérifie si la condition est remplie. Si oui, la ligne suivante est traitée, sinon, le reste de la commande est executée.
C'est en fait "l'inverse" de la commande IF.

Exemple :

```
If not "%ScoreJoueur"=="%ScoreNormal" echo Vous etes un nul
```

IF EXIST *Fichier*

Vérifie l'existence du fichier désigné. Si il existe, le reste de la ligne est traité, sinon on passe à la ligne suivante. Ce type de commande peut-être aussi utilisé sous la forme "If not exist", dans ce cas le reste de la commande est traité que si le fichier n'existe pas. Il est aussi important de noter que vous n'êtes pas obligé d'utiliser des guillemets puisque le paramètre représentant le fichier ne peut-être nul.

Exemple :

```
If exist c:\Autoexec.bat Copy autoexec.bat autoexec.old
```

IF ERRORLEVEL

Vérifie le numéro de message d'erreur.
Des commandes MS-DOS renvoient un numéro spécial au fichier batch en cas de problème ou d'erreur, désigné par ERRORLEVEL. ERRORLEVEL vaut toujours 0 si aucune erreur ne s'est produite. MS-DOS exécute le reste de la ligne si ERRORLEVEL est **égal** ou **supérieur** à la valeur spécifiée.

ATTENTION. Si vous devez tester plusieurs valeur de ERRORLEVEL, testez –les de la plus grande à la plus petite (ex : if errorlevel 255.. if errorlevel 100… if errorlevel 50…, etc) car comme dit ci-dessus, MS-DOS exécute le reste de la ligne si ERRORLEVEL est égal ou supérieur à la valeur spécifiée.

Il n'y a pas besoin de signe "=" entre errorlevel et le nombre représentant sa valeur.

Exemple :

```
Format a:
If errorlevel 3 echo Vous avez annule FORMAT par Ctrl+C !
```

Utilisation avec la commande GOTO :

Nous avons utilisé la commande IF pour introduire des questions dans les fichiers Batch. Il serait souhaitable maintenant d'utiliser plusieurs commandes en fonction du resultat de la question.

Voilà comment nous allons procéder :

```
If "<1>" == "<2>" Goto Suite
Commande 1
Commande 2
:Suite
Commande 3
```

Ainsi, si A=2, les commandes 1, 2 et 3 seront exécutées , sinon, la commande 3 sera exécutée et les commandes 1 et 2 évitées.

```
If not "%1"=="/?" Goto Suite
Echo Voici l'aide de ce programme
Echo Bla bla bla bla
:Suite
Echo Pour commencer, pressez une touche
Pause
```

Dans le cas si dessus, si le paramètre envoyé au batch n'est pas "/?"; les commandes après "Suite" sont executées. Sinon, le texte d'aide est affiché.

A retenir…

- IF permet d'agir différemment suivant qu'une condition est vraie ou fausse
- IF n'accepte qu'une seule commande à sa droite, c'est pour cela que la commande "Goto" sera régulièrement utilisée, pour exécuter ou non certaines parties du Batch.
- Il y a différentes formes du IF : IF, IF EXIST, IF errorlevel et IF NOT qui peuvent être combinées.

6°) Boucles

Après avoir fait connaissance avec une technique de la programmation des sauts inconditionnels (Goto), en voici une autre.

Nous allons créer un petit batch qui va afficher successivement les chiffres 1 à 4.

Ecrivez le fichier batch suivant :

```
@echo off
for %%A in (1 2 3 4) Do Echo C'est le nombre %%A
```

Ce fichier Batch contient une boucle FOR…DO. A quoi sert-elle ? Tout d'abord, %%A est utilisé seulement en tant que nom de variable. Cette variable prend alors toutes les valeurs de la liste spécifiée entre les parenthèses : dans notre cas, %%A prend donc successivement les valeurs 1, 2, 3, et 4. Les valeurs constituant la liste doivent être séparées entre elles par des espaces, des virgules, ou des points-virgules.
Ensuite, la commande qui suit immédiatement est exécutée avec la valeur prise par la variable %%A. Dans notre cas, on verra à l'écran le message "C'est le nombre" suivi de la valeur de la variable à chaque exécution de ECHO.
Un autre intérêt de cette commande est que les éléments de la liste peuvent-être des noms de fichiers. Ainsi il est possible d'exécuter une seule commande pour plusieurs fichiers. Vous pouvez donc afficher à l'écran plusieurs fichiers à la fois avec un seule commande qui est TYPE :

```
FOR %%A IN (AUTOEXEC.BAT CONFIG.SYS) DO TYPE %%A
```

Vous pouvez aussi utiliser les caractères génériques, par exemple :

```
FOR %%A IN (*.TXT *.BAT ) DO TYPE %%A
```

Tous les fichiers texte et Batch s'afficheront à l'écran.

A retenir…

- Une boucle FOR… DO…permet d'utiliser une variable prenant successivement toutes les valeurs d'une liste prédéfinie, et l'utilisation de cette variable dans des commandes DOS ou Batch standard.

7°) La compilation

PowerBatch vous permet de compiler un fichier Batch, c'est à dire de le transformer en un exécutable binaire Windows (.exe ou .com).

Un exécutable présente en effet plus d'avantages qu'un fichier Batch : vitesse d'exécution plus élevée, code source "protégé", format binaire inaltérable, etc…

La compilation n'est pas assurée par PowerBatch, elle est effectuée par un logiciel indépendant appelé "Bat2exec". Ce dernier n'est pas compatible avec toutes les commandes DOS et Batch, par conséquent, testez bien le fichier compilé avant de le distribuer pour éviter toute mauvaise surprise. Par exemple, la commande "CHOICE", n'est pas supportée par le compilateur.

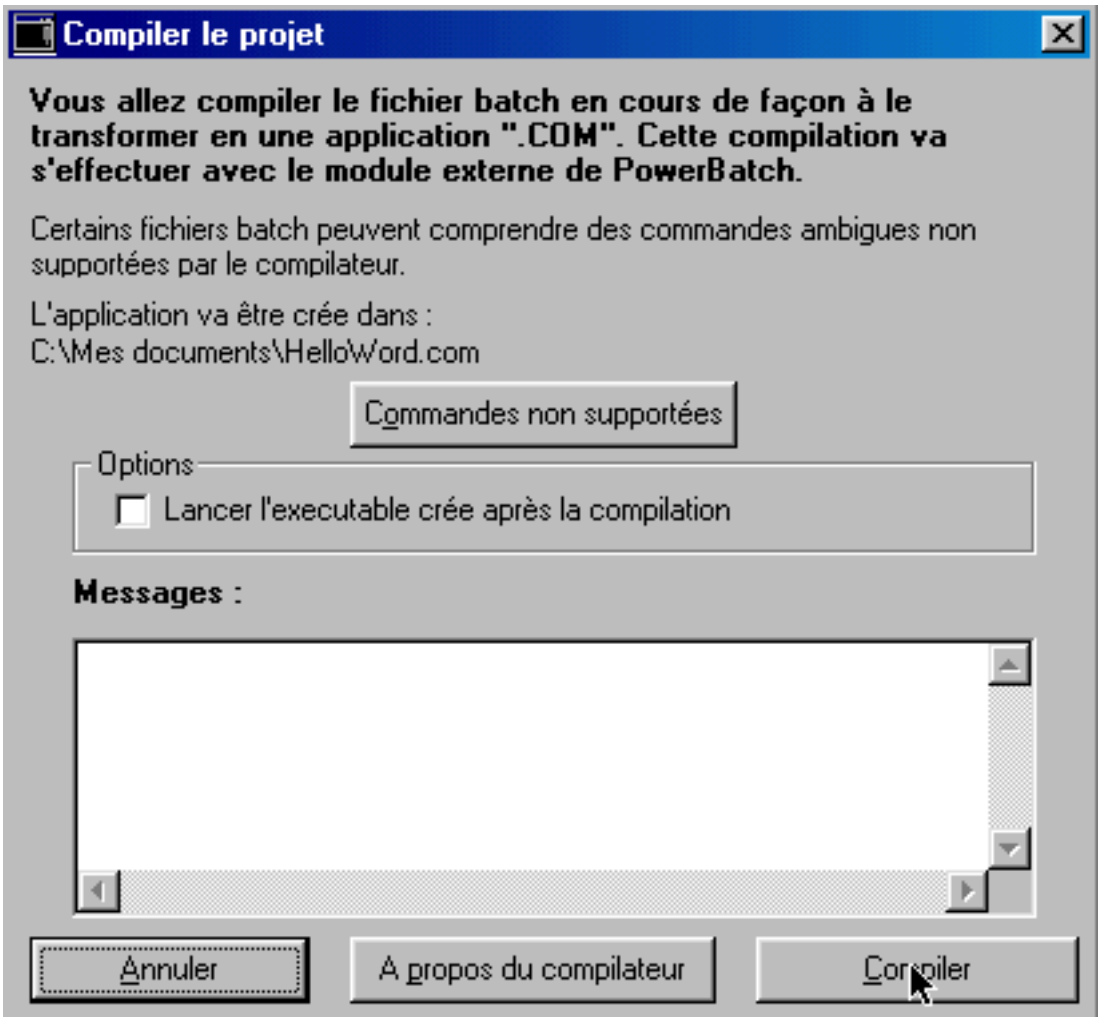
Compiler un fichier

1°) Créez ou ouvrez un fichier Batch. Dans notre exemple, il contient simplement :

```
@echo off
echo Bonjour, pressez une touche…
pause
```

2°) Choisissez la commande "Compiler" dans le menu "Fichier", puis nommez le fichier qui va être crée.

Une nouvelle fenêtre apparaît :

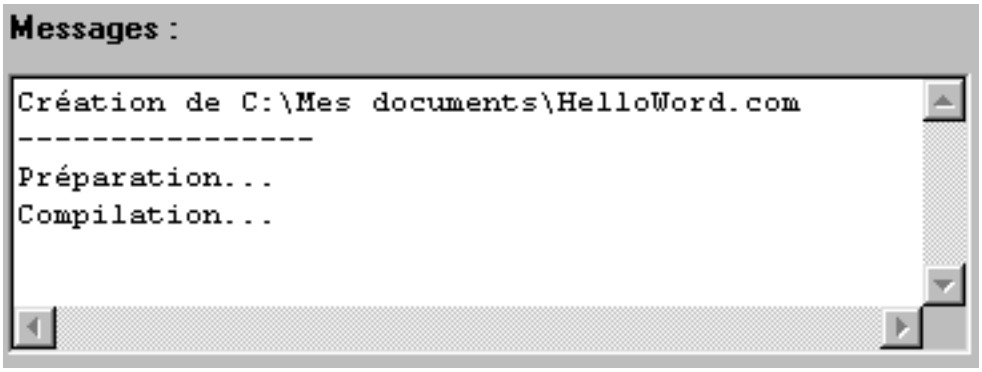


Cliquez sur "Compiler" pour compiler le fichier Batch.

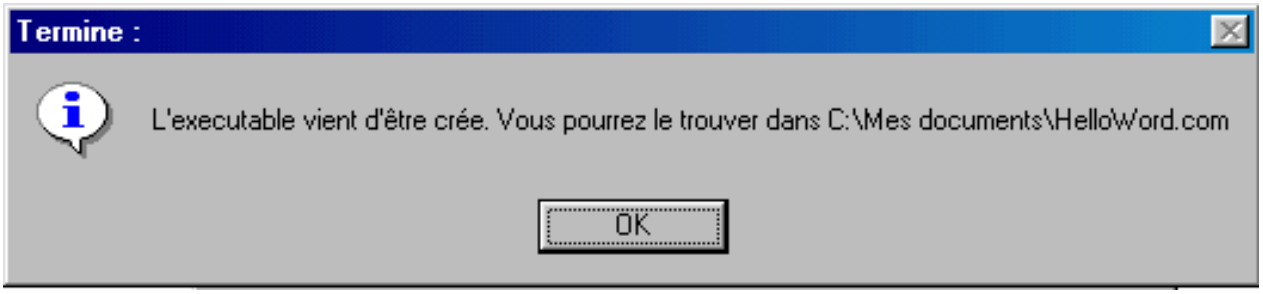
Un fichier ".com" sera créé, résultat du code compilé par Bat2exec.

Compilation sans erreur

Si toutes les commandes ont été supportées, et que Bat2Exec n'a rencontré aucune erreur, PowerBatch affiche les messages :



… puis la boite de dialogue :



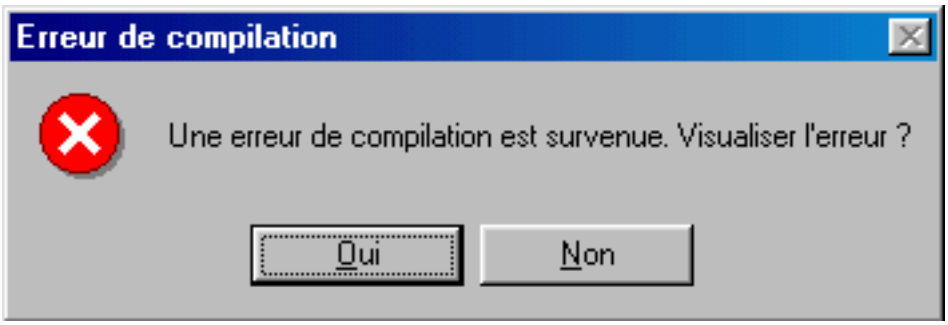
Compilation avec erreur

Si Bat2exec rencontre des erreurs lors de la compilation, il lui sera impossible de créer le fichier ".com".

Si par exemple, nous introduisons une erreur dans notre code…

```
@echo off
echo Bonjour, pressez une touche…
Goto Bonjour
```

… (en effet, il y a un "Goto" qui pointe vers un label inexistant) et que nous essayons de compiler le code, nous obtenons ce message d'erreur :



Cliquez sur "Oui" pour que Bat2exec vous montre l'erreur qu'il a rencontrée, dans notre cas, on a :

```
BAT2EXEC 1.5 (c) 1990, 1991 Ziff Communications Co.
PC Magazine ■ Douglas Boling

Error in line 5
Label BONJOUR not found
```

Il ne vous reste plus qu'à reprendre votre code pour le corriger.
Utilisez la barre d'état situé sous la zone de texte de PowerBatch qui affiche la ligne en cours pour détecter d'ou vient l'erreur d'après le n° de ligne transmis par Bat2exec.

A retenir…

- Compiler un fichier Batch, c'est transformer des commandes Batch en du code machine directement executable par l'ordinateur : c'est donc transformer un fichier texte en un programme binaire d'extension ".com".
- La compilation est assurée par un programme autonome nommé "Bat2exec".
- Si des erreurs de compilation surviennent, la création du programme est interrompue et ne peut – être recommencée que si cette erreur est corrigée.

8°) Les bordures

L'art de "faire" les bordures dans un fichier Batch est très apprécié des connaisseurs et des novices : quoi de plus esthétique d'encadrer un texte de cette facon :

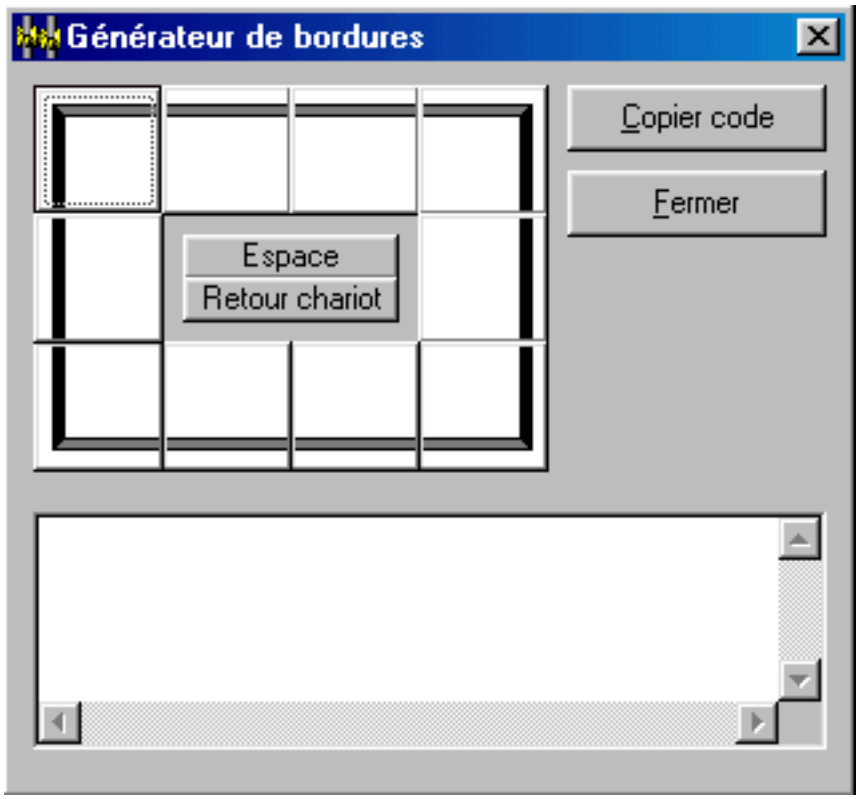
Pour cela, MS-DOS utilise tous les caractères "spéciaux", c'est pour cela que dans le chapitre 1 nous vous avons conseillé d'éviter d'utiliser les caractères accentués tels que "é,ç,à" etc…

En réalité, voilà ce qu'il faut entrer dans un Batch pour faire cette bordure :

```
@echo off
echo ÉÍÍÍÍÍÍÍÍ»
echo ºBonjour º
echo ÈÍÍÍÍÍÍÍÍ¼
```

Au lieu d'entrer ces caractères à la main, utilisez l'assistant créateur de bordures de PowerBatch (Dans le menu "Outil").

Voici comment se présente l'assistant à son lancement :



Comme vous le voyez, cet assistant comporte une sorte d'"encadrement" constitué de plusieurs images représentant un cadre fictif.

Il vous faudra en fait cliquer sur la case représentant la bordure voulue pour qu'elle apparaisse dans la zone de texte de la fenêtre.

Pour créer la bordure haute (1 coin haut/gauche, 8 traits horizontaux, et 1 coin haut/doit), correspondant à la ligne :

… vous cliquerez 1 fois sur la case :

…8 fois sur la case :

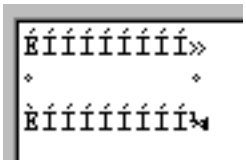
et 1 fois sur la case :

Ensuite, il vous faut aller à la ligne.
Cliquez sur "Retour chariot" pour créer une nouvelle ligne.
Nous devons donc entrer la seconde ligne pour créer une bordure ressemblant à :

Cette bordure est constituée de : 1 ligne verticale, vous cliquerez donc 1 fois sur la case représentant un trait vertical, 8 espaces, vous cliquerez donc 8 fois sur la case "espace", puis 1 trait vertical, vous cliquerez donc 1 fois sur la case représentant un trait vertical.
Ensuite, allez à la ligne pour créer la dernière ligne de la bordure :

Cette bordure est constituée de : 1 coin bas/gauche, 8 traits horizontaux, un coin bas/droit : vous utiliserez donc les cases de l'assistant appropriées.

La zone de texte de l'assistant contient maintenant :

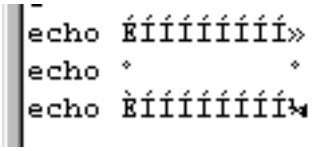


Ne nous préoccupons pas pour l'instant du texte "Bonjour" à intercaler dans la bordure. Notre bordure à proprement parler est maintenant créée. Pour l'insérer dans le fichier Batch, cliquez sur :

La fenêtre de l'assistant se ferme, et le code est maintenant copié dans le presse-papier.

Collez ce code à l'endroit voulu dans le batch à l'aide de la commande "Coller" du menu "Edition".

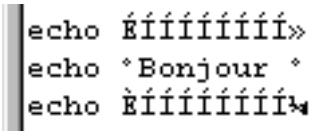
Dans votre Batch, vous avez maintenant :



```
echo É╔╔╔╔╔╔╔╔»
echo º        º
echo È╔╔╔╔╔╔╔╔╝
```

Pour afficher le fameux "Bonjour", il ne vous reste plus qu'à l'intercaler dans la seconde ligne, en veillant à ce que les bordures verticales (représentées ici par des "º") restent alignées avec les coins (ici `É, >>,` È,et `1/4`)

On a donc maintenant notre bordure :



```
echo É╔╔╔╔╔╔╔╔»
echo ºBonjour º
echo È╔╔╔╔╔╔╔╔╝
```

Testez le fichier… Et le résultat est bien celui attendu !

Par conséquent, utilisez l'assistant créateur de bordures pour encadrer des textes automatiquement, si vous ne souhaitez pas entrer manuellement les caractères spéciaux affichant les bordures.

Note : Il existe d'autres styles de bordures non supportées par l'assistant de PowerBatch. Dans ce cas vous devrez les rentrer manuellement.

9°) Écriture dans les fichiers Batch
Le caractère de redirection ">"

Ecrire dans des fichiers

Vous pouvez écrire dans des fichiers, à l'aide de commande Batch.
Nous avons dit dans le chapitre 1 que la commande ECHO servait en fait à "écrire" quelque chose quelque part. Pour l'instant, nous nous sommes contenté d'"écrire" sur l'écran, mais rien ne nous empêche de le faire sur le disque.
Nous allons aussi utiliser les chevrons (">" ou "<") comme caractères de redirection.
Vous devez veiller au nombre de chevrons, et à leur sens, en effet, la sortie sur le fichier en dépendra.

Écriture en mode "ajout" (Append)

Ce mode permet d'ajouter des données sans écraser celles qui étaient inscrites précédemment dans le fichier.

Nous allons utiliser 2 chevrons, orientés vers la droite, qui pointent vers le nom de fichier à utiliser :

```
Echo Texte à écrire>>c:\texte.txt
```

Ainsi, tout le texte compris entre "Echo" et les ">>" sera écrit dans "c:\texte.txt".

• Si le fichier n'existe pas, il sera créé et les données y seront inscrites sans générer d'interruptions ou d'erreurs, sauf si le ou les répertoires le contenant n'existent eux-même pas.
• Le texte à inscrire sera ajouté à la fin du fichier
• Une nouvelle ligne sera créée dans le fichier à chaque fois que vous appellerez la commande

Exemple pratique : vous souhaitez exécuter le programme StartServer.exe situé dans C:\www, au démarrage de votre ordinateur. Il vous suffira d'écrire :

```
Echo C:\www\StartServer.exe>>C:\Autoexec.bat
```

La commande DOS c:\www\StartServer.exe sera inscrite dans Autoexec.bat et lancée à chaque démarrage.

Écriture en mode "Ecrasement" (Output)

Contrairement au mode d'ajout, le mode d'écrasement efface toutes les données inscrites précédemment dans le fichier, puis inscrit la ligne transmise.

Nous allons utiliser **1 seul chevron**, orienté vers la droite, qui pointe vers le nom de fichier à utiliser :

```
Echo Texte à écrire>c:\texte.txt
```

Comme précédemment, tout le texte compris entre "Echo" et les ">>" sera écrit dans "c:\texte.txt".

• Si le fichier n'existe pas, il sera créé et les données y seront inscrites sans générer d'interruptions ou d'erreurs sauf si le ou les répertoires le contenant n'existent eux-même pas
• Le contenu du fichier sera automatiquement effacé**. Toutes les données seront perdues** et remplacées par le texte entre "echo" et ">"

Par exemple, vous souhaitez sauvegarder le contenu d'une variable (Ici %CPT%) dans le fichier "score.dat" situé dans C:\MonJeu\Scores\ :

```
Echo %CPT%>>C:\MonJeu\Scores\Score.dat
```

Comme nous l'avons dit plus haut si le fichier n'existe pas, il sera créé et les données y seront inscrites sans générer d'interruptions ou d'erreurs sauf si le ou les répertoires le contenant n'existent eux-même pas. Par conséquent, si les dossiers "MonJeu" et "Scores" ne sont pas présent sur le disque au moment de l'exécution de la commande, MS-DOS affichera un message d'erreur et le fichier ne sera pas crée. Il va également de soi que la variable CPT doit-être précédemment définie, en utilisant une commande de la forme **Set *CPT***=20000 .

Ecrire le résultat d'une commande dans des fichiers

Vous pouvez inscrire le résultat d'une commande DOS dans un fichier, avec les deux modes décrits plus haut ("Écrasement" et "Ajout").

Pour cela, vous n'avez qu'à supprimer "Echo", et remplacer le texte à écrire dans le fichier par une commande MS-DOS.

Par exemple :

```
dir c:\*.*>>c:\listing.txt
```

Le contenu du disque C:\ sera inscrit en mode "rajout" dans le fichier listing.txt

La redirection vers "nul"

"Nul" représente un périphérique virtuel inexistant. Utilisé avec ">" et ">>", il permet d'"écrire" le résultat de commande vers rien du tout, c'est à dire, en clair, de les masquer.

Par exemple :

```
Pause>Nul
```

Le texte normalement affiché par la fonction pause ("Presser une touche pour continuer") n'est pas affiché, seule la fonction demeure (l'utilisateur doit presser une touche pour que le déroulement du programme continue).

Note : NUL peut être aussi utilisé pour tester si un lecteur existe, avec une commande de la forme `if exists g:\NUL faitquelquechose` , "if" testant si un fichier virtuel pouvant représenter n'importe quel élément en réalité sur le disque existe.

10°) Appel d'autres fichiers Batch

La commande CALL permet d'appeler un fichier Batch à partir d'un autre fichier batch. Après avoir traité le fichier batch appelé, le programme revient au premier fichier batch et à l'endroit précis où le fichier batch a été appelé.

Vous pouvez également appeler un fichier batch à partir d'un autre sans pour autant revenir au fichier batch de départ. Il suffit tout simplement d'appeler le fichier batch par son nom (ou son adresse) c'est à dire sans CALL.

Appel sans CALL

Vous pouvez appeler un fichier batch à partir d'un autre en utilisant son nom. Le résultat est que le batch appelé est traité, mais il est impossible de revenir au batch de sortie précédemment traité. On peut en quelque sorte parler de "liaison unilatérale".

Exemple :

```
C:\MesBatch\fichier.bat
```

Appel avec CALL

Un batch X appelle un batch A à un endroit précis. CALL a pour rôle de contrôler que MS-DOS remarque bien le "point de saut" et revienne dans le batch appelant après avoir traité le batch appelé.
Le Batch A est donc utilisé comme un sous-programme. Cette utilisation comporte un avantage majeur : on doit programmer une seule fois les routines batch et on peut ensuite les appeler le nombre de fois que l'on veut à partir de n'importe fichier Batch.

Exemple :

```
CALL c:\MesBatch\Routine1.bat
```

11°) Travail avec ERRORLEVEL

De nombreuses commandes MS-DOS revoient une valeur de retour différente de 0 quand une erreur se produit. Dans le fichier Batch, elle peut-être consultée à l'aide de la variable ERRORLEVEL. ERRORLEVEL 0 signifie qu'aucune erreur ne s'est produite. Si vous programmez en C des extensions pour MS-DOS, vous pouvez renvoyer des valeurs à l'aide de l'instruction **`return`**.

Cette valeur peut-être testée avec IF, mais attention, il y a un léger point à surveiller : **si la valeur de retour est SUPERIEURE OU EGALE au numéro indiqué** la commande est exécutée. Par conséquent, si vous avez plusieurs ERRORLEVEL à tester, commencez toujours par la plus grande, puis procédez par ordre décroissant.

Exemple : le fichier Batch suivant formate une disquette dans le lecteur A. Si une erreur se produit ou si le processus est interrompu avec CTRL+C, le fichier Batch renvoie un message d'erreur.

```
@echo off
format a:
if errorlevel 1 goto erreur
goto fin
:erreur
echo.
Echo Formatage impossible !
:fin
echo on
```

Second exemple. Remarquez que nous contrôlons toujours la valeur la plus élevée :

```
Echo off
Format a:
If errorlevel 4 goto erreur4
If errorlevel 2 goto erreur2
Echo Pas d'erreur, formatage effectué
Goto fin
:erreur4
echo Lecteur ou parametre non valable
goto fin
:erreur2
echo Formatage interrompu avec CTRL+C
goto fin
:fin
echo on
```

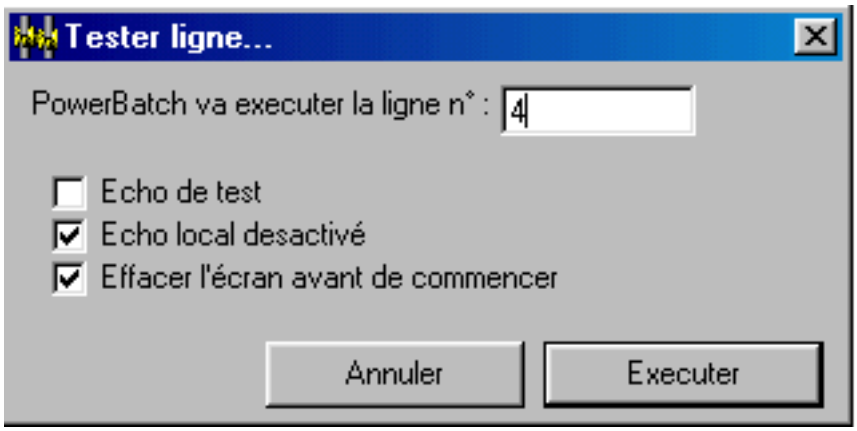
Toutes les commandes DOS ne renvoient pas des valeurs d'erreur. Les commandes concernées n'utilisent que certaines valeurs.

12°) 5 autres fonctions de PowerBatch

1°) Le test ligne, le test de bloc, le test pas à pas,

PowerBatch présente diverses possibilitées de test de vos fichiers batch :

• Le test ligne est obtenu en pressant la touche F8 (ou avec le menu Programme>Deboggage) :



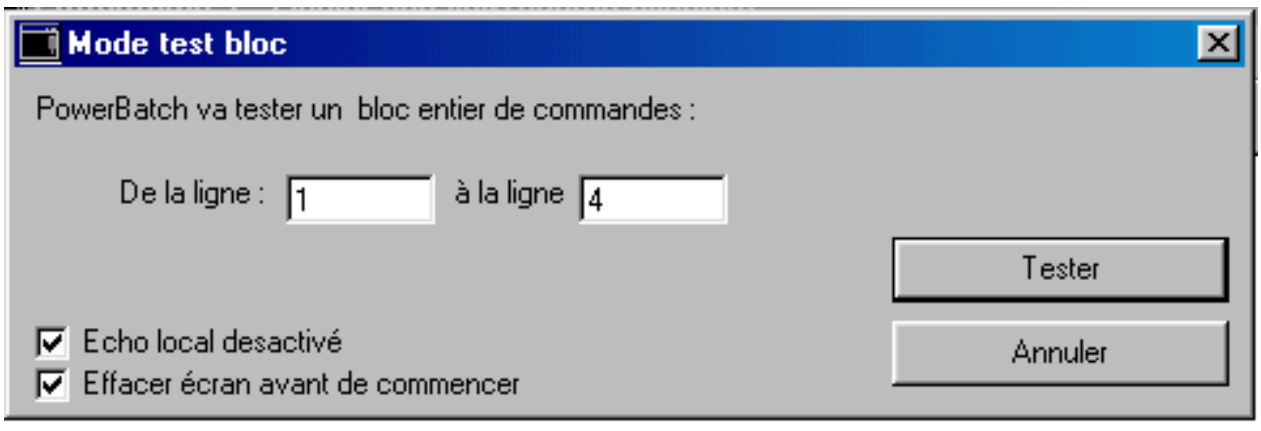
Cette fonction vous permet de tester une seule ligne de votre fichier.

Pour tester une ligne, vous devez entrer le numéro de la ligne dans la zone de texte de la fenêtre.

Vous pouvez automatiquement :

- Afficher un echo de test, rappelant quelle ligne va être exécutée
- Désactiver l'écho local
- Effacer l'écran avant de commencer (commande CLS)

• Le test bloc est obtenu en pressant la touche F9 (ou avec le menu Programme>Deboggage) :



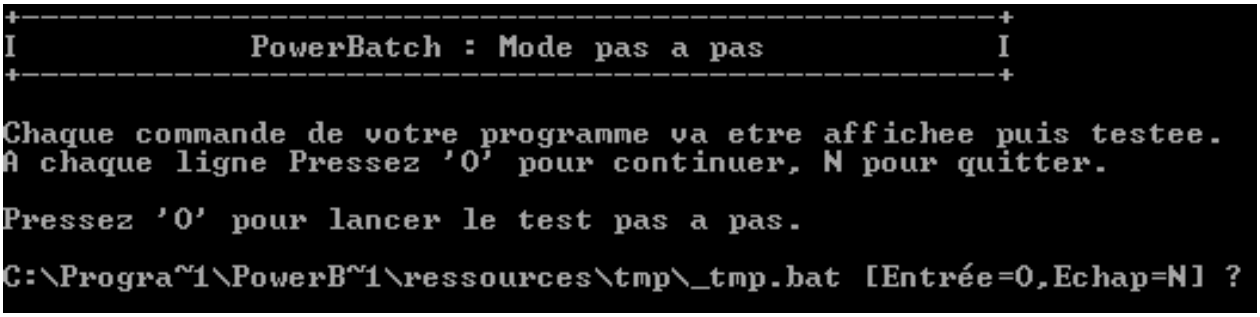
Cette fonction vous permet de tester un bloc de commandes, de la ligne X à la ligne Y.

Entrez la ligne de départ dans la première zone de texte, puis la ligne d'arret dans la seconde zone.

Vous pouvez automatiquement :

- Désactiver l'écho local
- Effacer l'écran avant de commencer (commande CLS)

Le test pas à pas est obtenu en pressant la touche F7 (ou avec le menu Programme>Deboggage) :

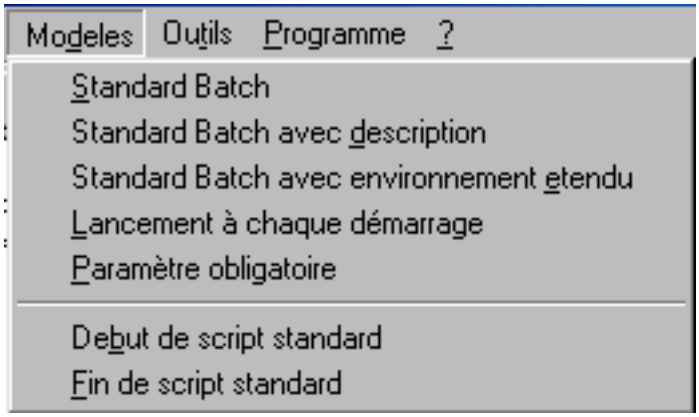


Ce mode vous permet de tester chaque ligne de code. Vous pourrez voir quelle ligne déclenchera les erreurs, quelle valeur prendra les variables, etc…

Vous devrez presser la touche "O" ou "N" à chaque ligne, la touche "O" permettant de continuer le texte, la touche "N" de le stopper.

Pour commencer un test pas à pas vous devez obligatoirement presser la touche "O" de votre clavier dans la fenêtre DOS qui s'affichera.

2°) Les modèles Batch



PowerBatch présente 5 modèles complets de Batch.
Le plus utile est sans doute "Standard batch avec environnement étendu" car il permet de parer les erreurs dus à un espace d'environnement insuffisant.

De plus, y sont ajoutés deux "macros", ensemble de commandes régulièrement tapées, soit en début de batch, soit en fin de batch : le *début de script* insère **@echo off** pour désactiver l'écho local, puis **cls** pour effacer l'écran, et le fin de script rétabli l'écho à l'aide de la commande **echo on**.

3°) L'assistant XCOPY

La commande XCOPY est une commande DOS permettant d'effectuer des copies avec plus d'options de la commande COPY. L'assistant XCOPY a été introduit dans PowerBatch afin de vous aider à faire des copies de fichiers en utilisant des paramètres et des options valides.

Vous pouvez lancer cet assistant à l'aide du menu "Outils".

4°) La commande CHOICE

La commande CHOICE permet d'introduire des entrées clavier dans un batch. Attention, il n'est pas question d'entrer du texte, mais juste d'appuyer sur une touche et d'agir en fonction de la touche préssée, pour faire des messages du style : `Pour continuer, pressez C, pour quitter, pressez Q`

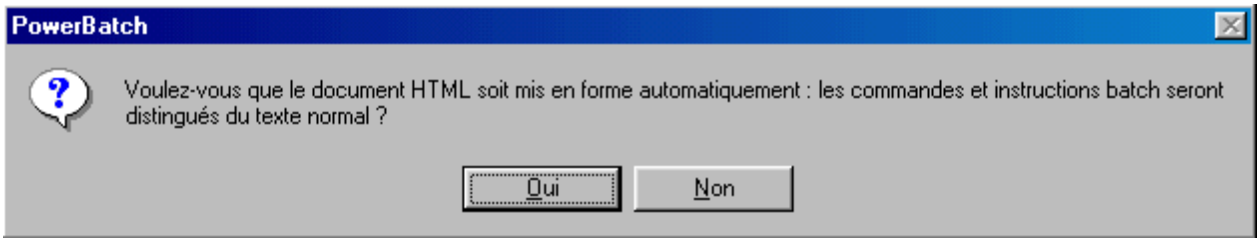
Cet assistant vous permet d'utiliser cette commande de façon optimale, et configure rapidement des messages avec entrée clavier.

Vous pouvez lancer cet assistant à l'aide du menu "Outils".

5°) Le convertisseur HTML

Le HTML est un langage "universel" de description de document, utilisé notamment sur Internet pour bâtir des pages Web.
Ne croyez pas que le HTML est indissociable du Web, et que l'utilisateur doit être connecté sur Internet pour lire ce type de fichiers : il sera très bien lu hors-ligne, chez une personne ne possèdent même pas Internet.
L'avantage est que ce langage est lu par différents logiciels (navigateurs) sur la majorité des systèmes d'exploitation (Windows, MacOS, Linux…). Utilisez donc la conversion dans ce format si vous voulez exporter le code d'un fichier Batch sur un autre ordinateur sans altération du code; ou pour le transférer par Internet, sur un site web, ou par e-mail.

PowerBatch permet de mettre en relief le code converti (les commandes DOS seront distinguées, les commentaires mis en italique etc.). Il suffit de répondre positivement à ce message :



Le fichier sera crée, et une nouvelle icône apparaîtra, ressemblant à celle – ci :

Double-cliquez sur l'icône pour lancer le navigateur associé aux fichiers HTML (en général Microsoft Internet Explorer)

Pour finir…

J'espère que vous avez suivi ce bref tutoriel avec plaisir, et que cette initiation à la programmation en langage Batch ne vous a pas parue trop compliquée.

Je vous conseille de trouver des d'autres didacticiels et documents présentant des astuces de programmation et d'autres sujets non traités dans ce document. D'autre part, si vous souhaitez utiliser des commandes d'extensions à MS-DOS pour vos batchs (saisie clavier, opérations logiques, tirages de nombres aléatoires, etc…) je vous invite à télécharger le toolkit Batch à partir de http://www.astase.com4.ws .

Si vous remarquez des erreurs, ou pensez que des compléments sont nécessaires, merci de me contacter (rabusier@aol.com)

Bonne continuation !

Rabusier