Tutoriel sur la programmation Batch

Par Adrien REBOISSON – http://www.astase.com

Seconde Édition

Attention : ce tutoriel a été à l'origine écrit pour les personnes possédant Windows 95, Windows 98, ou Windows Me. L'intégration sous Windows NT étant différente (le terme « Windows NT » regroupant Windows NT, Windows 2000, ou Windows XP), nous vous conseillons si vous possédez Windows NT de lire ce tutoriel **sans oublier l'annexe II** qui détaille les différences majeures entre l'intégration Batch sous les distributions standard et les distributions NT de Windows.

Table des matières

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 via le site web (http://www.astase.com)
La version la plus récente de ce manuel sera toujours publiée sur http://www.astase.com

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 transfert 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 comme 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 votre premier Batch

Un fichier Batch étant à la base un fichier texte, vous pouvez créer vos batchs avec n'importe quel éditeur de texte.
Attention : un fichier batch est un fichier texte « brut » (*.txt) sans formatage particulier (gras, italique, souligné). Personnellement, je trouve inutile d'utiliser un programme aussi lourd que Microsoft Word pour écrire un minuscule bout de fichier ASCII ! Préférez le bon vieux Notepad (bloc-note Windows) ou le simpliste mais néanmoins utile EDIT sous DOS, si vous êtes puriste…

Si vous débutez ou si vous souhaitez gagner du temps, je ne peux que vous conseiller d'utiliser PowerBatch, un programme que j'ai moi-même écrit et qui facilite le test et le déboggage de vos batchs. Ce tutoriel est plus axé sur l'écriture de vos Batchs avec PowerBatch, mais le langage étant le même que vous utilisiez Notepad ou PowerBatch, ce tutoriel pourra tout de même vous aider dans le cas où vous ne souhaiteriez pas utiliser mon application.
PowerBatch est disponible sur http://www.astase.com.

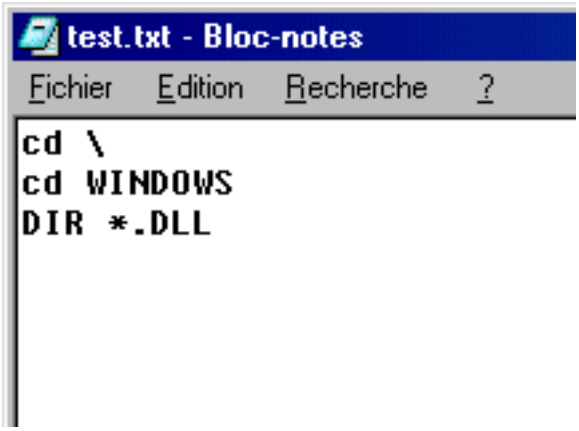
Prenons le cas simpliste ou vous souhaitez lister les fichiers DLL du répertoire de Windows (C:\WINDOWS). Vous entreriez sous DOS :

```
C:\Chemin> cd \ [Entrée]

C:\> cd WINDOWS [Entrée]

C:\WINDOWS> DIR *.DLL [Entrée]
```

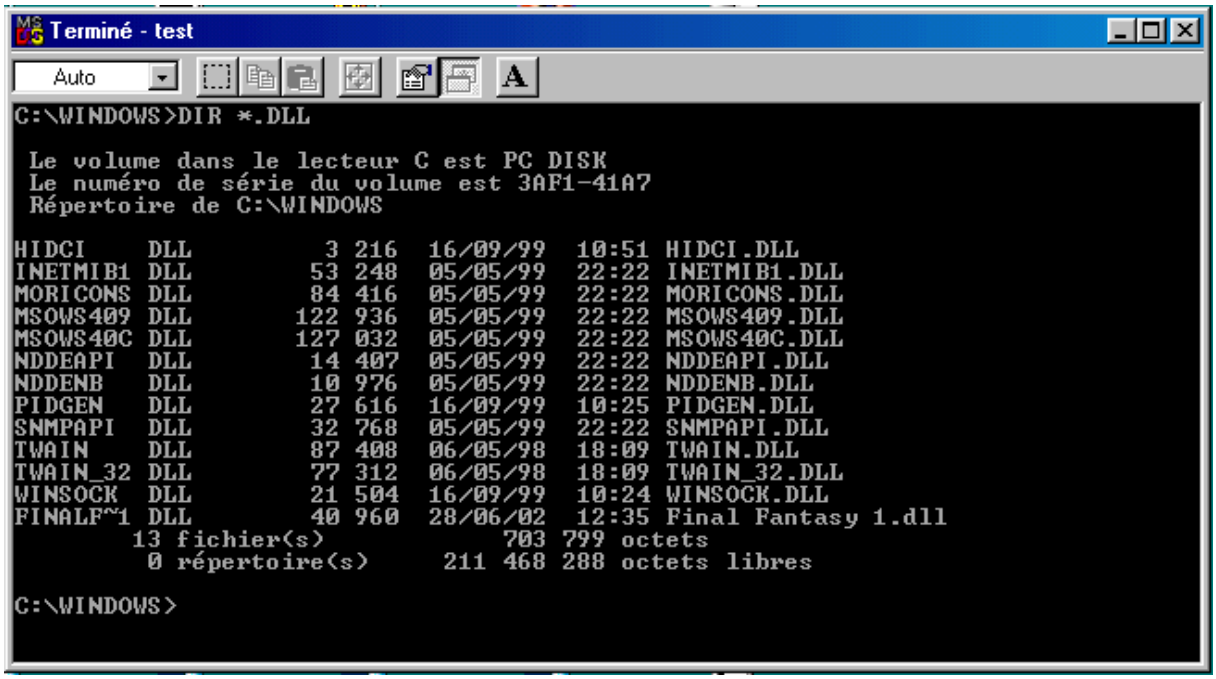
Il vous suffit de taper dans un fichier texte les commandes précédentes, comme si vous les entriez sous le prompt *MS-DOS* :



Facile, non ? La rédaction de votre Batch est terminée.
À notre stade, nous avons créé un fichier texte – il faut maintenant le transformer en Batch. Là aussi, pas trop de difficultés : il suffit de changer son extension en `.bat` sous Windows. Dans notre exemple, *test.txt* devient donc *test.bat.*

Maintenant, il ne vous reste plus qu'à tester votre Batch !

Un fichier Batch s'exécute sous MS-DOS. Si vous êtes sous MS-DOS, il vous suffit d'appeler le fichier dans la ligne commande. Si vous êtes sous Windows, c'est plus simple puisqu'il suffit de double-cliquer sur l'icône du Batch pour que celui-ci s'ouvre dans une console MS-DOS (Une parfaite simulation du DOS par Windows), et ne s'exécute aussitôt :



Que s'est-il passé ? Que vous soyez sous Windows ou sous MS-DOS, c'est le même processus : le fichier batch est transmis à l'interpréteur COMMAND.COM, qui analyse chaque ligne du batch et exécute les commandes rencontrées.

Comme vous pouvez le constater, la différence avec d'autres programmes Windows est de taille : alors que les exécutables « normaux » sont des fichiers binaires constitués de macro-instructions destinés directement au processeur, les fichiers batchs contiennent les commandes DOS « telles qu'elles », ce qui explique la nécessité d'un interpréteur pour « traduire » les commandes au processeur ;c'est le rôle de COMMAND.COM – il n'y a donc pas besoin de compiler le code.

Si la fenêtre se ferme automatiquement sans vous laisser le temps de visualiser son résultat, rajoutez la commande « PAUSE » à la fin de votre batch. Même si vous ne comprenez pas encore cette commande, sachez que celle-ci demande une validation clavier avant de terminer le programme, ce qui vous laissera le temps de visualiser le contenu de la fenêtre.

Comme dit auparavant, les Batchs ont été conçus et s'exécutent sous MS-DOS. Le terme MS-DOS rappelle pour certains la « préhistoire » de l'informatique moderne et peut conduire à un désintérêt total en pensant que le DOS est un système obsolète et désormais « enterré », qui ne mérite de nos jours plus grand intérêt.

D'un côté c'est vrai : depuis l'avènement des OS graphiques comme Windows, MS-DOS est en phase de déclin et est de plus en plus abandonné, tant par les développeurs que les utilisateurs. Certains utilisateurs des dernières versions de Windows ignorent même jusqu'à l'existence du DOS !

Et pourtant, si le DOS à part entière n'a pas grand intérêt, la cohabitation MS-DOS/Windows est toujours de nos jours intéressante. Cherchez un langage simple pour automatiser vos opérations sous Windows : le système Batch est là !
Même sous Windows, et ce grâce au DOS, vous pourrez automatiser la plupart de vos opérations. Après la lecture de ce tutoriel, vous pourrez même je l'espère, développer de petits programmes complets et complexes.

Pour aborder un tout autre point – certains peuvent se poser la question : Les fichiers batchs peuvent être dangereux pour mon système ? A priori, les fichiers Batch ne sont pas plus dangereux que n'importe quelle application DOS ou Windows. En plus, il suffit d'éditer le fichier pour visualiser son code et se rendre directement compte de la dangerosité potentielle de certaines commandes. Il convient toutefois de rester prudent avec les fichiers batchs de provenance non sûr (en pièce jointe transmise par e-mail par exemple). Etant donné qu'une seule commande peut détruire votre système (ou l'endommager fortement le cas échéant), je vous conseille d'éditer tout fichier Batch non « sûr » avant l'exécution afin de vous rendre compte par vous-même de la dangerosité de ceux-ci.

3°) Utilisation de PowerBatch

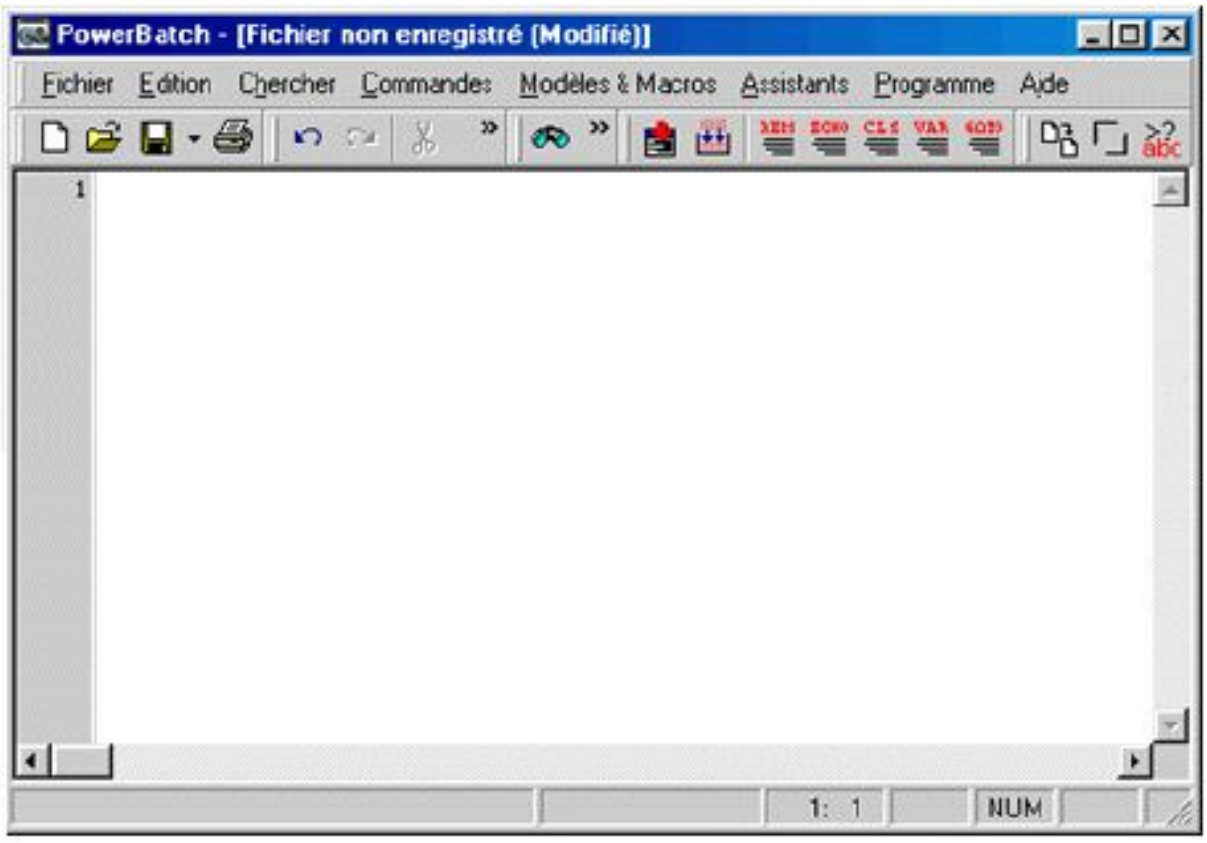
Réalisons le Batch écrit plus haut dans le bloc note avec l'application PowerBatch.
Démarrez PowerBatch en double-cliquant sur son icône ou via les raccourcis implantés dans le menu Démarrer.

PowerBatch présente de nombreux avantages pour la réalisation de fichiers Batchs dont :

- Affichage des différentes lignes dans le Batch,
- Test rapide des fichiers – envois simplifiés des paramètres,
- Support du format Batch natif (distinction des commandes, enregistrement direct en *.bat),
- Insertion automatisée des commandes les plus fréquentes,
- Nombreux assistants de création de commandes,
- Déboggeur intégré,

etc…

Au lancement, PowerBatch affiche la fenêtre suivante (légèrement différente selon les versions) :



Dans la zone de texte, entrez les commandes précédemment stockées dans le fichier texte créé avec Notepad :

```
CD \
CD WINDOWS
DIR *.DLL
```

Sauvegardez votre Batch via le menu Fichier. Pour tester votre Batch, il vous suffit d'aller dans le menu « Programme », « Tests », « …Du fichier courant », ou bien de presser simplement la touche F9.

Le programme est testé dans un répertoire temporaire, puis PowerBatch vous redonne la main.

• Pour tester vos commandes, utilisez le sous menu « Console MS-DOS » du menu « Programme ». Cela affiche la console DOS afin que vous puissez tester vos commandes avant de les introduire dans vos batchs.

• PowerBatch propose un support graphique au compilateur Bat2Exec. Pourquoi compiler vos batchs, puisque COMMAND.COM est là pour les interpréter ? Pour protéger votre code, accroître la vitesse d'exécution, utiliser le format binaire moins altérable que le format ASCII… Le compilateur Bat2exec transforme donc vos batchs en programme binaires d'extension « *.COM ». Pour plus d'informations, consultez le chapitre « Compilation de vos Batchs » dans ce manuel.

• PowerBatch comporte 3 assistants, placés dans le menu « Assistants » :

  • *Le créateur de copies* vous permet de créer le code correct pour une utilisation de XCOPY ou XCOPY32,

  • *Le créateur de bordures* vous permet de créer vos encadrements dans vos batchs. Un chapitre de ce tutoriel est consacré à cette fonction.

  • *Assistant d'entrée clavier* permet de créer des demandes au clavier, ou de créer facilement des menus.

• Vous pouvez également utiliser l'assistant de recherche des erreurs (dans le même menu). Celui-ci détectera les sauts incorrects dans votre Batch (ceux-ci seront abordés plus loin dans le batch).

• Le menu « Modèle et Macro » contient des exemples et des modèles des fichiers Batchs les plus utilisés pour vous faire gagner du temps.

• Le menu « Commandes » regroupe des commandes DOS ou Batch fréquentes et vous permet ainsi de les insérer dans vos Batchs en conservant leur syntaxe propre.

• Le menu « Fichier » contient des formats d'exportation particuliers pour vos batchs, permettant de conserver la mise en forme de ceux-ci dans d'autres applications ou dans d'autres systèmes d'exploitation. Une section de ce tutoriel est consacrée à cette fonction.

L'utilisation de PowerBatch est détaillée dans le fichier d'aide accessible dans PowerBatch en pressant F1. Ce tutoriel va maintenant se concentrer sur la programmation Batch en tant que tel, et suppose que vous savez maintenant Créer, Ouvrir, Sauvegarder, et Tester vos batchs avec PowerBatch. Vous pouvez maintenant créer vos fichiers avec PowerBatch ou Notepad – mais dans ce dernier cas vous pourrez passer certains chapitres traitant de certaines fonctionnalités de PowerBatch.

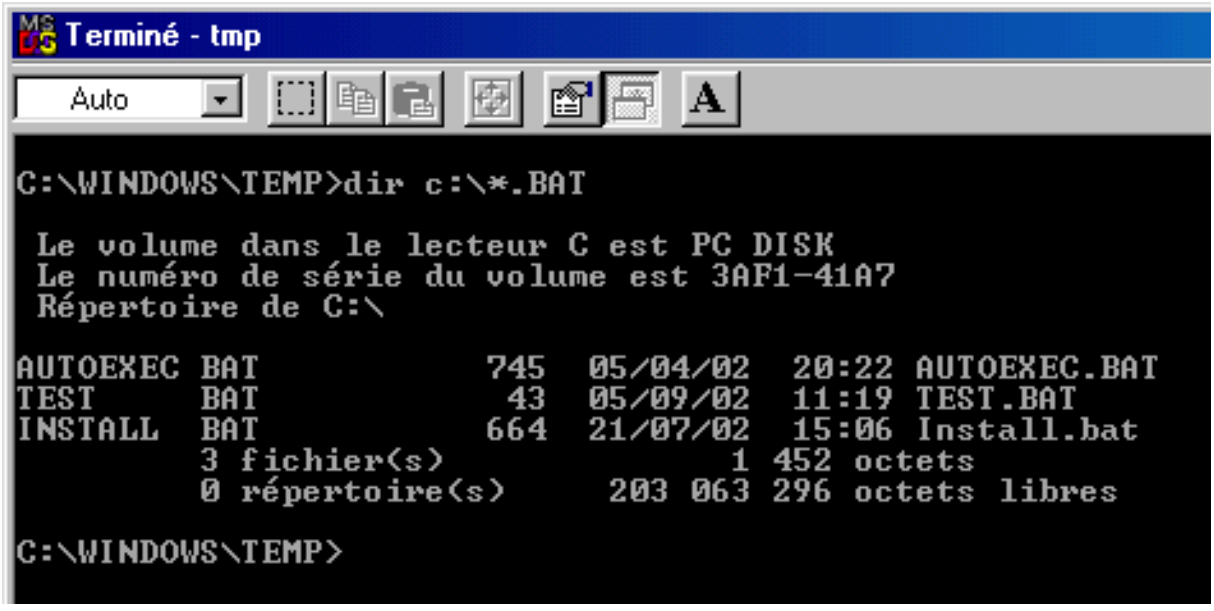
2°) Les bases de la programmation Batch

Vous vous doutez bien qu'un fichier Batch n'est pas constitué uniquement de commandes DOS. Il existe, et c'est l'intérêt de celles-ci, des commandes* spécifiques aux fichiers batch. En voici les plus simples.

L'écho

Le terme « Echo » a plusieurs significations. Selon qu'il est utilisé, on peut parler de commande de sortie, propriété d'affichage ou commande de désactivation ou d'activation.
C'est une notion simple mais très souvent utilisée dans la programmation Batch.

Lorsque vous crééz un fichier Batch, vous entrez des commandes MS-DOS à la suite. Lorsque vous exécutez le batch, il apparaît à l'écran sous une forme un peu spéciale : il est affiché d'abord *la commande* comme si vous l'aviez tapée vous même sous DOS, puis son *résultat.* Par exemple, si vous entrez un batch simpliste contenant l'unique commande « `DIR c:\*.BAT`», et que vous l'exécutez , il apparaît à l'écran un résultat du type:



En haut de l'écran apparaît la commande, comme si vous l'aviez vous même tapée (DIR c:*.BAT), puis son résultat en bas (le catalogage des fichiers Batch à la racine du disque C:\ - celui-ci étant le mien, vous n'aurez pas le même résultat que cette capture d'écran !)

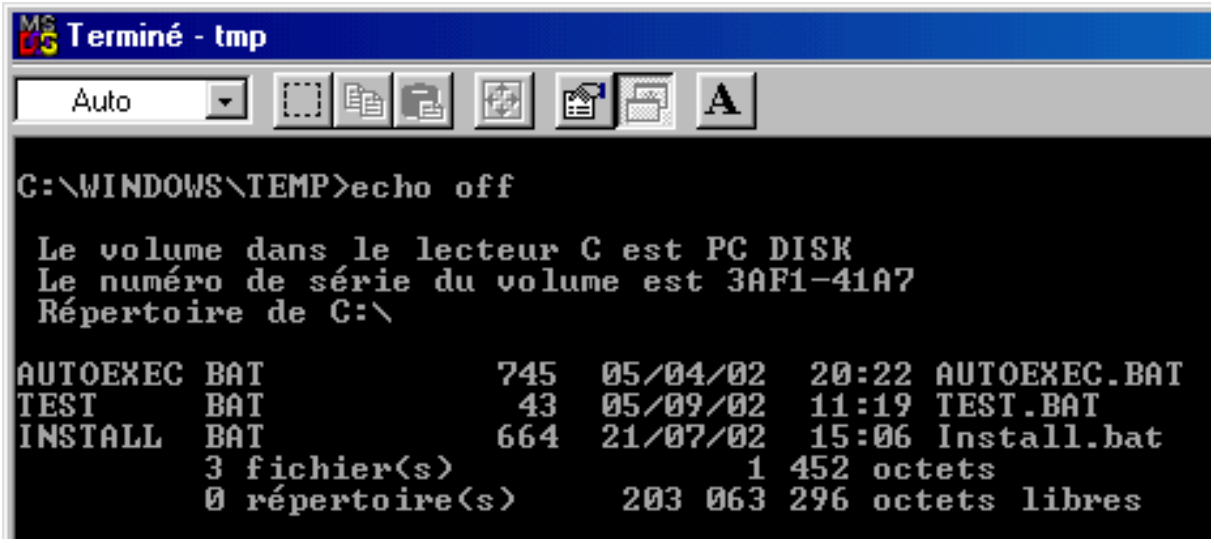
Maintenant imaginez que vous souhaitez *simplement afficher le résultat de la commande*. Il va donc falloir trouver un moyen de *masque*r la ligne affichant la commande avant son exécution.

Pour arriver à cela, vous pouvez utiliser la commande `echo off` qui désactive l'écho. C'est en effet par « echo » que la fonction d'affichage avant exécution est appelée.

Si vous réécrivez votre batch en lui ajoutant à son début « echo off », cela donne :

```
Echo off
Dir c:\*.bat
```

Et si vous l'exécutez, vous aurez un résultat du style :



Seul le *résultat* de la commande « dir C:*.BAT » a été affiché a l'écran.

Dans votre Batch, l'écho est désactivé lorsque COMMAND.COM rencontre `echo off`. Ce dernier n'est pas réactivé tant que la commande inverse : `echo on`, n'est pas rencontrée.

Pourquoi, puisque l'écho est désactivé via la commande « `echo off` », cette dernière est-elle justement affichée en haut de l'écran ? Et bien tout simplement car MS-DOS affiche d'abord, et exécute ensuite. Rencontrant « `echo off` », il l'affiche, puis désactive après l'écho. Nous verrons ultérieurement comment contourner cette fonction.

Un autre moyen pour désactiver l'echo est de faire précéder la ligne dont seul le résultat doit-être affiché du signe « @ » .

Ainsi, vous pourriez transformer le batch

```
Echo off
Dir c:\*.BAT
```

…en …

```
@dir *.BAT
```

Bien sûr, si notre batch comportait plusieurs lignes, il faudrait rajouter autant de « @ » devant chaque ligne qu'il aurait de lignes ou l'on veut désactiver l'écho. Autant, dans ce cas-là, utiliser une seule fois en début de Batch la commande « `echo off` ».

Cette notion d'écho local doit être saisie, car la majorité des programmeurs Batchs préfèrent le désactiver par économie de place et souci de compréhension (cela évite que beaucoup trop de lignes incompréhensibles pour le néophyte soient affichées à l'écran) : vous rencontrerez donc souvent les commandes précédemment citées.

Un dernier point sur l'écho : la commande « @echo off ». Il s'agit d 'une combinaison du signe « @ » et de « echo off ». Quel est son interêt ? Et bien cela évite d'afficher « echo off » en début de batch comme il apparaît dans la dernière capture d'écran. La désactivation de l'écho est elle même masquée via « @ ». Vous rencontrerez donc très souvent cette combinaison.

Vous pouvez réactiver l'écho à tout moment dans le Batch en insérant la commande inverse « `echo on` ».

Commenter son code

Comme dans tout langage de programmation, il est essentiel de commenter son code :

- Cela accroît la clarté du code et le rend compréhensible par n'importe quel collaborateur,

- Cela vous permet vous-même de vous repérer dans votre code si celui-ci est eu peu complexe,

- Cela facilite votre relecture si par exemple vous n'avez pas travaillé sur un code depuis longtemps.

Pour introduire un commentaire, utilisez l'instruction `REM`, puis entrez une ligne de texte, par exemple :

```
REM Catalogage de C:\TEXTES
DIR C:\TEXTES
```

La ligne précédée de REM ne sera pas exécutée, mais tout de même affichée à l'écran si l'écho est activé. Dans tous les cas, la présence de commentaires ne gène en aucun cas l'exécution de votre code ; cela alourdit néanmoins légèrement votre fichier.

Afficher du texte à l'écran

Il peut être utile, dans certains cas, d'afficher un texte à l'écran, par exemple pour informer l'utilisateur de ce que « fait » le Batch.

Vous utiliserez la commande `echo` . Encore elle !
Oui, mais là son usage est différent. **Si cette instruction n'est pas suivie de « off » ou de « on », elle permet d'afficher un texte à l'écran.**
En réalité, ECHO est utilisé pour faire sortir tous types de données : par défaut le texte est envoyé à l'écran, mais vous pouvez l'envoyer sur l'imprimante, dans un fichier etc… Pour l'instant, considérons simplement que ECHO permet d'afficher un texte.

Si vous souhaitez par exemple afficher « Bonjour » à l'écran, rien de plus simple puisqu'il vous suffit d'entrer :

```
Echo Bonjour
```

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

Si l'écho local est activé, le texte sera affiché deux fois : une fois précédé de la commande d'affichage « `echo` », lorsque MS-DOS affichera la ligne, une fois sans « `echo` », lorsque MS-DOS exécutera la ligne. Une bonne raison de désactiver l'écho local par un simple « `@echo off` » en début de batch !

Voici un petit batch qui mêle commentaires, messages à l'écran et commandes DOS :

```
@echo off
REM Désactive l'écho local
CD C:\WINDOWS
REM Va dans le rep. De Windows
Echo Les fichiers WFM vont être effacés !
DEL *.WFM
Echo Terminé !
```

Afficher des lignes vides : attention, vous ne pourrez pas afficher des lignes vides en entrant juste « `echo` ». En effet, dans ce cas, vous découvrirez un 3è aspect de `echo` : s'il n'est suivi de rien, MS-DOS affiche l'état de ECHO : ON s'il est *activé*, OFF s'il est *désactivé*. Par conséquent, il existe une « combine » pour afficher une ligne vide : il faut entrer « `echo.` »

Exemple :

```
@echo off
echo Tentative infructueuse pour afficher une ligne vide
REM MS-DOS va afficher l'état de ECHO car on a oublié le « . »
après « echo » :
Echo
Echo  Affiche maintenant une ligne vide :
Echo.
Echo Termine.
```

… qui donne :

Résumé :

Vous avez découvert 3 aspects de ECHO :

- Pour afficher du texte, on utilise « ECHO » suivi du texte à afficher,
- Pour activer ou désactiver l'écho local, on utilise « ECHO ON » ou « ECHO OFF »
- Si l'on veut afficher l'état de l'écho (actif / non actif), on utilise « ECHO » sans autre paramètre.

Sous MS-DOS ou sous Windows, l'écho est activé ou désactivé uniquement pour le batch en cours d'exécution. Cette propriété « locale » est propre à chaque batch. L'écho étant par défaut activé, si vous souhaitez le désactiver pour tous vos batchs, il faudra entrer dans chacun d'eux « `echo off` » ou « `@echo off` »

Les caractères spéciaux

S'il y a bien un point important sur lequel il faut être vigilant, c'est bien sur celui des caractères spéciaux. Ceux-ci désignent notamment tous les caractères accentués de notre langue – n'oubliez pas que le DOS a été à l'origine conçus par des américains - , ainsi que les signes %,<,>, |, car ils correspondent à des sigles bien précis qui pourraient entraîner de multiples erreurs s'ils étaient insérés sans raison particulière dans un batch. Il suffit simplement de ne pas les utiliser dans vos batchs.

Prenons l'exemple d'une phrase bien accentuée :

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

.. celle ci donnera à l'écran :

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

Dans la version 5.5 et les versions ultérieures de PowerBatch, les accents sont automatiquement remplacés par leur équivalent DOS, ce genre de problème ne peut donc plus arriver.

Effacer l'écran

Pour effacer l'écran (du moins le texte de la console DOS si vous travaillez sous Windows). Il existe une commande bien connue par les habitués du Basic : CLS, pour Clear The Screen – effacer l'écran.

Imaginez une commande produisant des résultats textuels importants : pour ne pas « noyer » l'utilisateur sous un flot de caractères quelconques, faites suivre l'instruction incriminée de CLS. Le texte sera immédiatement effacé, le prompt DOS revenant au coin haut gauche de l'écran.

Si vous travaillez sous DOS, il peut être utile de commencer tous vos batchs par CLS pour effacer l'écran sans doute déjà occupé par de nombreuses lignes de commandes.

Sous Windows, si un batch se termine par CLS, Windows aura tendance à fermer automatiquement la fenêtre (surtout sous les versions NT et XP de ce système). Cela est logique : pourquoi continuer d'afficher à l'écran une ligne vide, « nettoyée » par CLS, donc exempt d'informations pour l'utilisateur ?

Faites une pause !

À la différence de CLS, instruction utilisée pour nettoyer l'écran des informations devenues inutiles, il existe une commande permettant de laisser des données à l'écran tant que l'utilisateur ne presse pas une touche. L'exécution est ainsi interrompue, laissant le temps à l'utilisateur de visualiser le résultat d'une commande, un texte d'information, un message d'avertissement, etc.

Il suffit pour cela d'entrer l'instruction `pause` dans votre Batch. Lorsque l'interpréteur rencontre l'instruction « pause », il apparaît à l'écran :

Si vous souhaitez simplement attendre l'appui d'une touche sans afficher le message classique généré par MS-DOS, il suffit d'entrer :

```
Pause>nul
```

Le rôle de « `>nul` » sera abordé ultérieurement. Comparez-le simplement à votre stade à un « trou noir » supprimant tous les messages normalement émis par la commande à sa gauche.

Alors qu'un CLS en fin de fichier Batch force Windows à fermer la console DOS, un `pause` en fin de Batch fait tout le contraire : il empêche la fermeture de la fenêtre tant qu'une touche n'est pas pressée. Il peut donc être utile de terminer vos batchs par cette instruction si vous possédez des OS comme Windows XP qui ont tendance à fermer automatiquement les batchs une fois exécutés.
Pour sortir d'un Batch en pause (ou en exécution), pressez la combinaison de touche CTRL+C (Break)

Avec le contenu de ce chapitre, vous pouvez rédiger de petits batchs utilisant les notions suivantes :

- Contrôle de l'écho local par « `@echo off` » ou « `echo on` »,
- Effacement de l'écran par `CLS`,
- Attente de l'appui sur une touche avec `pause`,
- Textes divers en utilisant `echo` .

… tout cela, bien sûr, abondamment commenté par des « `REM` » !

NOTES POUR CE CHAPITRE :

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

3°)Variables et paramètres

Comme dans tous langage de programmation, il existe un moyen de stocker des informations dans un emplacement de la mémoire de votre ordinateur. A cet emplacement est associé un nom que vous définissez, afin de manipuler facilement ces données.

Généralités

Vous rencontrerez des variables sous plusieurs formes :

- Lors de la lecture : c'est-à-dire lorsque vous examinerez et utiliserez le contenu d'une variable,

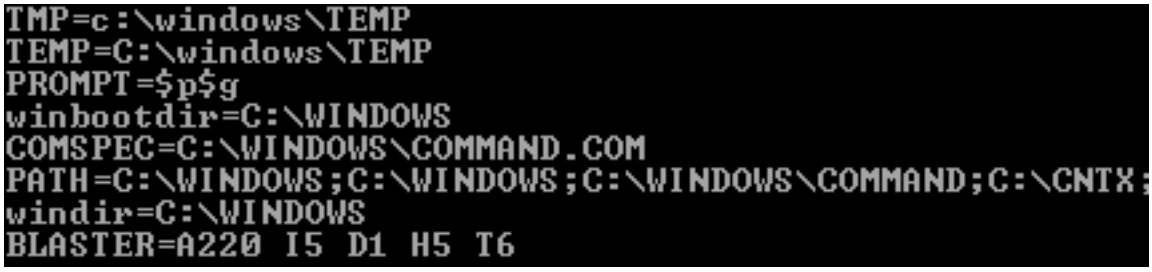
- Lors de l'écriture : c'est-à-dire lorsque vous modifierez le contenu d'une variable.

Dans tous les cas, la casse, c'est-à-dire la différence majuscules/minuscules n'est pas discriminatoire pour le DOS : il considère par exemple la variable Abc et aBC comme identiques. Par compte, il est sensible à la casse du *contenu* des variables. C'est-à-dire que lorsque vous aurez à tester le contenu d'une variable – si vous cette variable se nomme par exemple `Nom` et contient `frederic` – MS-DOS détectera une différence si le test se porte sur `FREDERIC` puisque le premier est en minuscules et le second en majuscules.

Variables d'environnement

Une *« variable d'environnement »* correspond au nom du seul type de variable que l'on peut utiliser pour stocker des données. Elle représente une valeur accessible n'importe ou et n'importe quand dans l'environnement DOS.
Vos variables sont stockées « temporairement » et détruites à la fin de votre Batch. Pour créer des variables permanentes (affichées ci-dessous par SET), vous devrez les insérer dans Autoexec.bat (Sous Windows NT, vous ferez : Panneau de configuration > Système > Avancé > Variables d'environnement)

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



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).

Si vous définissiez et utilisiez une variable dans un de vos Batch, tant que celle-ci ne serait pas détruite ou que le Batch ne sera pas terminé, elle serait affichée par `set` et apparaîtrait à l'écran.

Il est important de savoir que le contenu de ces variables est détruit une fois l'ordinateur éteint, la fenêtre ou la session DOS terminée. Il faut donc, si ces variables doivent être présentes à chaque session et si elles ne sont pas automatiquement déclarées par Windows, les définir dans Autoexec.bat (qui est lui lancé à chaque démarrage).

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

Introduction a la création de variables

Nous allons créer des variables, auxquelles nous assignerons des valeurs.
A la création de celles-ci, il faudra toujours spécifier une valeur initiale. Ainsi, à la variable « `Dossier4` » peut être affecté la chaîne de caractère : « `C:\WINDOWS\COMMAND\EDB\MAKECAB` ».

Utilisez la commande SET, qui requiert la syntaxe suivante :

`Set` **`NomDeVariable`**`=`**`Valeur`**

Lorsque l'interpréteur rencontre `SET`, il analyse ensuite le nom de la variable à sa droite, puis les données à droite du signe égal. Il associe ensuite un espace de votre mémoire vive à la variable crée, remplissant cette mémoire réservée par les données trouvées. Pour identifier cette variable, il lui associe enfin un nom, celui rencontré à la droite de `SET` (c'est quand même plus sympa pour vous de manipuler un nom qu'une adresse mémoire – du style 2200 :5F60 !)

Par exemple, vous souhaitez associer à la variable `SysVer` le texte « Windows 95 ». Vous entrerez donc :

```
Set SysVer=Windows 95
```

Lire et utiliser les variables

Pour lire le contenu d'une variable, on l'encadre de deux « % ». Ces deux signes indiquent à l'interpréteur qu'il doit remplacer le nom de la variable par son contenu. Si la variable n'existe pas, aucune erreur n'est affichée, mais la variable est remplacée par une chaîne de caractères nulle.

Soit « `nom` » une variable contenant un nom à afficher. On pourrait utiliser :

```
Echo Bonjour, %nom% !
```

Si la variable « nom » contient une valeur, celle-ci remplace donc dans le batch la chaîne « %nom% ». Si par exemple, cette variable contient le texte « Mathieu », alors, l'interpréteur « verra » la ligne suivante :

```
Echo Bonjour, Mathieu !
```

… et affichera à l'écran « Bonjour, Mathieu !».

Cette variable peut-être insérée n'importe ou dans votre batch, combinée à n'importe quelle commande. Si la variable « `rep` » contient une adresse du type « `C:\DOSSIER1` », vous pourrez l'utiliser avec d'autres commandes comme :

```
@echo off
cls
echo Création du repertoire %rep%
mkdir %rep%
echo %rep% a ete cree.
```

Ce qui donnera pour l'interpréteur les lignes suivantes, qu'il exécutera de suite (si `rep` contient bien « C:\DOSSIER1 ») :

```
@echo off
cls
echo Création du repertoire C:\DOSSIER1
mkdir C:\DOSSIER1
echo C:\DOSSIER1  a ete cree.
```

Faites bien attention si la variable sur laquelle vous travaillez n'est pas vide. Dans les 2 cas suivants pris en exemple, si « `nom` » est vide, cela génère des réactions plutôt inattendues :

```
Echo %nom%
```

Donne : « `ECHO est actif` », puisque l'interpréteur « voit » simplement la commande « ECHO » suivie de rien du tout – il affiche donc l'état de l'écho.

```
Mkdir %nom%
```

Donne « `Paramètre manquant` », puisque l'interpréteur voit encore la commande « `mkdir` » seule sans paramètres.

Manipulation des variables

Exemple de création de variable - Nous souhaitons définir une variable "VersionWindows" contenant « 98 SE »

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

```
Set VersionWindows = 98 SE
```

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

Pour tester 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. Nous pourrions l'utiliser dans un batch, par exemple :

```
Echo La version de Windows installee est %VersionWindows% .
```

Notons que :

1°)Cette variable ne sera détruite qu'à l'extinction de l'ordinateur ou à la fin de la session DOS

2°)N'importe quel autre programme peut lire, modifier ou réécrire sur cette variable.

Ajout de caractères – Vous avez associé à `SysVer` la chaîne « Windows 95 ».
Imaginons que vous souhaitiez ajouter « - 32 bits » à la fin de la chaîne afin que la variable contienne à un certain moment « Windows 95 – 32 bits ».

La première solution est simple : il faut redéfinir entièrement la variable dans le batch, sous forme, par exemple :

```
Set SysVer=Windows 95 – 32 bits
```

Il y a-t il une solution plus « élégante », plus pratique ? Bien sûr ! Vous pouvez entrer :

```
Set Sysver=%sysver% - 32 bits
```

Déconcertant ? Pour MS-DOS , c'est pourtant très simple : %sysver% sera remplacée par sa valeur (rappelez vous qu'elle contient déjà des données, en l'occurrence « Windows 95 ») avant que la ligne soit exécutée, cela donnera donc pour MS-DOS :

```
Set Sysver=Windows 95 - 32 bits
```

Cela a le grand avantage de fonctionner quel que soit le contenu de la variable. Par exemple, supposons que votre batch contienne la lettre d'une unité logique de votre PC dans la variable UNIT. Pour la transformer en chemin d'accès compréhensible, il faudra lui ajouter «:\ ». En effet, si UNIT contient « c », il faudra ajouter «:\ » pour avoir l'adresse valide « C:\ ». Avec cette technique, vous pourrez rajouter facilement «:\ » au contenu de la variable, quel que soit la lettre du lecteur. Vous pourrez donc taper :

```
Set unit=%unit%:\
```

Suppression d'une variable – De la même manière que `SET x=y` définit la variable x avec le contenu y, on peut être amené à vouloir détruire une variable (ce qui détruira naturellement sa valeur associée). Pour « nettoyer » la mémoire, ou bien vous serez peut-être amené à utiliser SET sous cette forme :

```
Set NomDeVariable=
```

Par exemple, pour supprimer de la mémoire la variable « Folder », on tapera :

```
Set Folder=
```

Variables remarquable – Certaines variables sont toujours présentes lorsque vous utilisez DOS alors que Windows est installé sur votre machine. C'est le cas de la variable *Windir*, qui pointe toujours vers le répertoire de Windows.

Imaginons par exemple que vous souhaitez effacer sur plusieurs postes le Démineur fourni avec Windows (le nom de fichier est « Winmine.exe », situé dans le dossier de Windows). Vous pourriez entrer dans un batch la ligne :

```
DEL C:\WINDOWS\WINMINE.EXE
```

Mais attention ! Si C:\WINDOWS est le répertoire d'installation par défaut, certaines configurations possèdent un répertoire différent, du style C:\WIN98, C:\WIN2K ou bien carrément C:\MONWINDOWS, etc… Pour éviter les erreurs, vous pourriez utiliser la variable *Windir* et écrire alors votre ligne :

```
DEL %WINDIR%\WINMINE.EXE
```

De la même façon, il existe d'autres variables pointant sur des répertoires ou des fichiers cruciaux de Windows (voir capture d'écran réalisée plus haut), en voici quelques-unes :

- TEMP, qui contient le chemin du répertoire temporaire de Windows,
- COMSPEC, qui pointe vers l'intérpréteur DOS (généralement COMMAND.COM – CMD.EXE sous NT),
- WINBOOTDIR, qui contient l'adresse du répertoire de démarrage de Windows,
- PATH, qui contient la liste des répertoires à utiliser pour la recherche des fichiers (voir plus bas).

Sous Windows NT, de nombreuses autres variables d'environnement sont présentes, relatives au nombre de processeurs, à l'utilisateur actif, au répertoire racine de l'utilisateur actif, etc…

Il est recommandé d'utiliser ces variables le plus que possible pour éviter des erreurs du style « répertoire introuvable » ou « fichier inexistant ».

Définition de variables de session

On appelle généralement variables « de session » des variables définies à chaque démarrage, et dont la présence est permanente jusqu'à l'extinction de l'ordinateur. Par exemple, le logiciel MASM (cité à titre d'exemple), rajoute de nombreuses variables de session contenant les dossiers d'installation, afin qu'il puisse traiter des fichiers dans ces dossiers sans manipuler directement ceux-ci.

Pour créer vos propres variables de session, éditez AUTOEXEC.BAT, et inscrivez la ligne de définition de variable correspondante. Par exemple, votre répertoire de travail est C:\ROMAIN\WIN32\ASM\DEBUG, et vous souhaitez stocker celui-ci dans la variable « Dossier » : vous entrerez donc dans Autoexec.bat :

```
SET Dossier=C:\ROMAIN\WIN32\ASM\DEBUG
```

À tout moment, vous pourrez utiliser « Dossier » pour désigner votre répertoire de travail. Un simple `CD %Dossier%` sous DOS vous placera dans le répertoire correspondant, sans avoir à taper une fastidieuse ligne de commande. Idem dans vos Batchs : vous manipulerez facilement ce répertoire dans toutes les opérations sur les fichiers.

Attention : pour que la variable puisse être définie à chaque démarrage, il faut impérativement que AUTOEXEC.BAT soit à chaque fois exécuté ! Sur certains PC, ce n'est pas le cas, et sur d'autres – les plus récents (XP) -, Windows ne connaît même plus de fichier de ce nom ! Dans ce cas, et sous Windows NT (NT4, 2000, XP) seulement, vous pouvez toujours définir des variables au démarrage par l'intermédiaire d'un clic droit sur le *poste de travail*, *propriétés*, puis *avancé*.

Paramètres

Votre batch est un programme. Comme tout programme, il peut recevoir des paramètres.
On appelle paramètre tous les arguments passés à un programme ou une commande.

Prenons l'exemple d'un formatage de disquette. Sous DOS, vous pourriez entrer :

```
Format a:
```

La simple ligne « `Format a :` » fait référence à un programme (en l'occurrence ici un exécutable nommé « `format.exe` »). Et « a: » est pour lui un paramètre, c'est-à-dire une information supplémentaire qu'il peut lire et traiter.

Les paramètres sont toujours séparés par des espaces. Par exemple :

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, etc…

Prenons l'exemple d'un batch qui doit effacer tous les fichiers TMP (*.TMP) d'un dossier. Si le batch est capable de recevoir comme paramètre le répertoire dans lequel il doit opérer, on pourra par exemple l'appeler sous DOS :

```
Batch1.bat C:\DEMO
```

… Pour qu'il supprime tous les fichiers TMPdu répertoire « DEMO ».

Vous comprenez maintenant l'intérêt des paramètres : cela introduit une sorte de modularité et une nouvelle souplesse dans la programmation. On peut créer des batchs qui sont des « sous-programmes », recevant des infos *via* les paramètres.

Il est important de noter que vous ne pouvez pas sous Windows envoyer directement des paramètres (le double-clic pour lancer un exécutable empêche toute frappe clavier). Cependant, vous pouvez ouvrir une console DOS dans Windows, vous placer dans le répertoire souhaité, puis lancer l'application en lui envoyant des paramètres par la ligne de commande DOS, ou éditer un fichier PIF pour cela – ces notions seront abordées ultérieurement.

Concrètement, un batch peut recevoir jusqu'à neuf paramètres. Ceux-ci sont stockés dans 9 variables notées %1 à %9. La variable %0 correspond quant à elle au chemin du batch.
Contrairement aux variables classiques, on observe 3 grosses différences : on ne peut pas les modifier, elles ne sont pas encadrées par 2 « % » (seulement précédés d'un seul « % »), enfin, elles sont spécifiques pour chaque batch.

Vous pouvez tester cela en créant le batch suivant :

```
@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 où vous n'envoyez aucun paramètre (vous lancez simplement le fichier), vous obtenez un résultat de ce type :

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

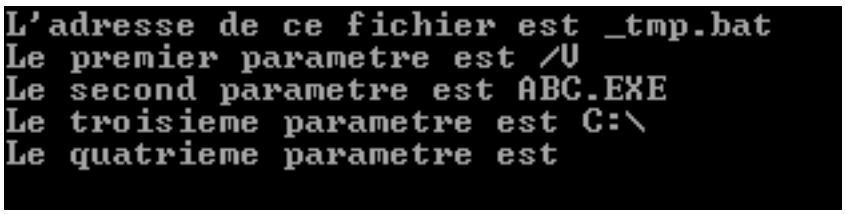
Sous DOS, vous pouvez lancer le fichier en le faisant précéder de son adresse, puis en envoyant les paramètres, par exemple :

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

Vous pouvez procéder plus facilement avec PowerBatch :
PowerBatch 5.0 à 5.5 : Entrez simplement ces paramètres dans la boîte de dialogue affichée juste avant n'importe quel test,
PowerBatch 5.6 et plus : Les paramètres sont définis dans la boîte de dialogue « Paramètres » du menu programme et envoyés à chaque test.

Résultat dans les deux cas :le fichier est exécuté avec les paramètres entrés.

Ce qui donne :



Ce qui est ici parfaitement logique.

On a dit que les paramètres étaient séparés par des espaces. En appelant le même fichier batch que celui précédemment crée et en lui envoyant la ligne « `Ceci est l'exemple même d'une longue ligne de paramètres` », le batch « saucissonnera » votre phrase en lui associant un mot pour chaque paramètre. C'est à dire que si votre phrase contient plus de neuf mots (seul 9 variables sont disponibles !), les autres seront totalement oubliés. Si, au lieu de traiter une phrase, on traite une adresse longue (du style « `C:\Mes Documents\Docs. Personnels\Impots 2002\Fiches 1er semestre\Fiche 1.xls` », on court à la catastrophe en éludant une partie importante de l'adresse !

Pour outrepasser cette limite, encadrez vos longues chaînes de guillemets. Ils inhibent les espaces de la phrase et permettent que celle-ci soit stockée dans une seule variable. MS-DOS étant compatible avec cette technique, vous pourrez manipuler vos fichiers avec n'importe quel exécutable.

Rien n'empêche d'associer à une variable d'environnement un paramètre. Cela permettra même de conserver la valeur du paramètre en dehors du batch qui l'a fixé. Exemple : supposons que votre batch reçoit comme premier paramètre l'adresse d'un dossier sur lequel il doit agir.

Vous pourrez entrer :

```
Set Dossier=%1
```

… Pour manipuler dans votre Batch ce dossier sous un nom plus explicite, ou pour permettre à d'autres Batch l'utilisation de ce répertoire.

Information : La commande ***shift*** permet de décaler le contenu des variables paramètre %x. %1 passe dans %0, %2 dans %1 et ainsi de suite... Et %0 est perdu. Avec ***shift*** *vous pouvez accéder au 11è paramètre* : qui est « transféré » dans %10. Vous pouvez ainsi appeler shift autant de fois que vous voulez, pour avoir accès aux paramètres supplémentaires.

La variable %PATH%

Comment un exécutable peut-être considéré comme une commande ? Une commande est par définition un "mot" que l'on peut entrer ou 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 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 :
>
> ```
> Superjeu.exe
> ```
>
> Vous pouvez simplement entrer :
>
> ```
> Superjeu
> ```
>
> … pour que MS-DOS "comprenne" que vous souhaitez lancer le programme "Superjeu.exe".

Mais où sont donc stockés ces commandes ?

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

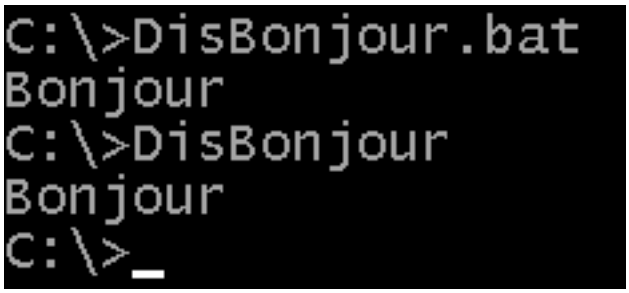
Si vous possédez un fichier .exe, .com ou .bat, et que vous souhaitiez l'établir en tant que « commande » DOS, copiez – le simplement dans ce répertoire.

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…



Résultat : 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.

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 :

```
C:\>path
PATH=C:\WINDOWS;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\CNTX
```

On peut voir que les répertoires d'accès sont au nombre de 3, et **séparés par des points-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 : 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 normale : cette variable étant modifiable, nous allons donc inclure un autre chemin d'accès dans cette variable.

Comme vous l'avez appris précédemment, on peut ajouter facilement des données à une variable en utilisant la forme suivante :

PATH=%PATH%;*CHEMIN_A_AJOUTER*

Rappelez vous que le point virgule n'est nécessaire que pour que le DOS puisse « séparer » les différents répertoires contenus dans PATH. Il est superflu dans une ligne ne définissant pas PATH.

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 une console DOS :

```
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 peut 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 (si, si, ce programme est assez spécial puisque même sous DOS, il marche ! – sauf sous Win XP)

Chez certains utilisateurs, de nombreux logiciels auront ajouté des répertoires à %PATH%, comme par exemple :

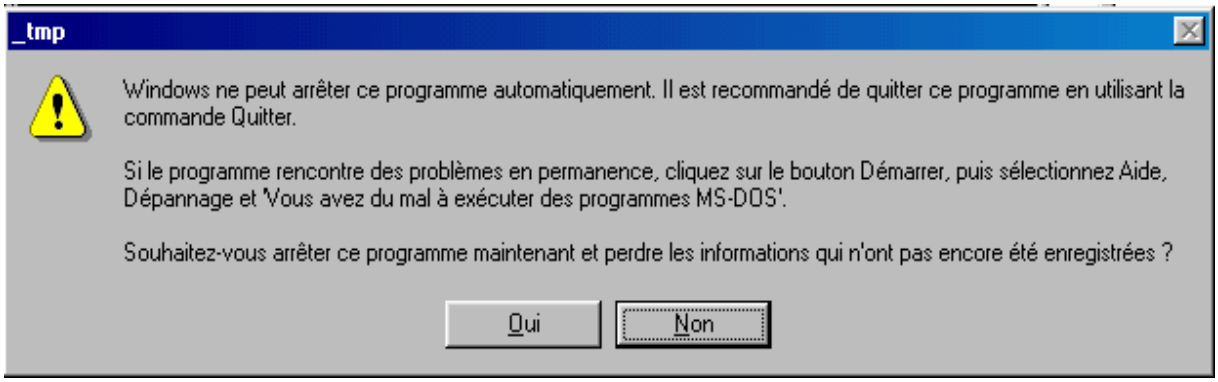
```
C:\>echo %path%
C:\BC4\BIN;C:\WINDOWS;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\CNTX;C:\PROGRA~1\BORLAND\
DELPHI5\BIN;C:\PROGRA~1\BORLAND\DELPHI5\PROJECTS\BPL
```

Pourquoi ? Imaginez vous concepteur d'un logiciel superbe, qui utilise MS-DOS pour lancer un programme auxiliaire, nommé par exemple RC.EXE, et installé quelque part dans Windows par l'utilisateur. Etant donné que l'utilisateur à pu via l'installateur, installer ce fichier n'importe ou dans votre machine, votre logiciel ne pourra pas l'appeler via une adresse du style « C:\W32IDE\BIN\RC.EXE », vu qu'il ne connaît pas le répertoire d'installation. Par compte, il suffit que l'installateur écrive dans AUTOEXEC.BAT l'adresse du dossier de RC.EXE dans le PATH pour que votre programme puisse le lancer en appelant simplement « RC.EXE ». Il en va de même pour l'utilisateur qui pourra appeler RC.EXE où qu'il soit dans votre disque, sans entrer manuellement l'adresse de l'exécutable.

4°) Saut, Boucles

Les instructions de sauts permettent de faire « boucler » votre programme, c'est à dire de répéter indéfiniment une commande ou un bloc de commandes. Ils permettent également, si on les couple à des conditions comme dans le chapitre suivant, d'éviter des blocs de code selon qu'un test s'est avéré juste ou faux.

Nous allons étudier dans ce chapitre la commande "Goto". C'est une commande de saut par défaut inconditionnelle, qui ne peut être arrêtée (ou à l'aide de commandes que vous ne connaissez pas encore), par conséquent vous allez être conduit 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 :

1) La première 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.

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

C’est comme un « télétransporteur » :

- Il faut un appareil de départ (Goto) qui propulse le voyageur…
- … vers le second point (Label), pouvant être placé avant ou après l’appareil de départ, qui reçoit le voyageur.

Si l’appareil d’arrivée est placé avant l’appareil de départ, *on obtient une boucle « sans fin »*, c’est-à-dire qui ne s’arrête jamais. Si l’appareil d’arrivée est placé après l’appareil de départ, *on a réalisé un « saut » dans le programme*, puisque si l’on se replace dans le cadre d’un batch les commandes entre les deux « appareils » sont évitées.

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", retourne 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. La différence majuscule/minuscule n’est pas prise en compte pour les labels et les Goto.

  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".

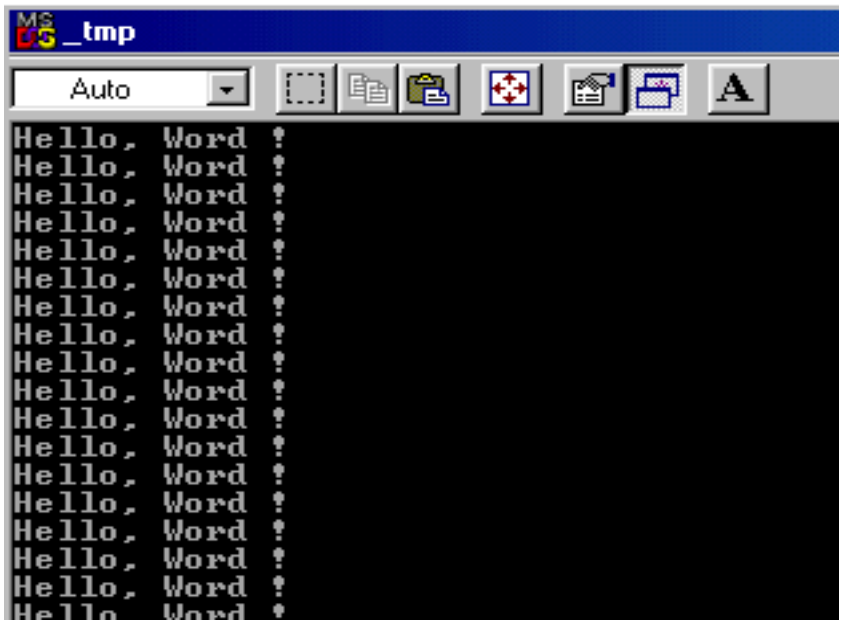
Allons – y pour une boucle infinie !
Nous voulons afficher "Hello, World !" en boucle.

Nous écrirons donc :

```
@echo off
:Start
echo Hello, World !
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

Reprenons : lorsque l'interpréteur rencontre « Goto », il analyse le nom du label qui le suit et le cherche dans tout le batch (il peut-être avant ou après l'instruction GOTO). Il sait qu'il doit le trouver précédé de « : », indiquant qu'il s'agit d'un point d'entrée potentiel du programme. Ici, il trouve le label avant le Goto : il se branche donc dessus et reprend son exécution, et retrouvant de nouveau le même Goto, le cycle recommençant indéfiniment…

Nous pouvons de la même façon éviter un bloc de code :

Commande 1
Commande 2
Goto Jmp1
Commande 3
Commande 4
Commande 5
:Jmp1
Commande 6
Commande 7

Quelles seront les commandes exécutées ? Réponse : la n°1, n°2, n°6, et n°7. Les commandes de 3 à 5 sont ignorées. Quel est alors l'intérêt de cette forme : puisque de toute façon les commandes 3, 4, et 5 ne seront pas exécutées, pourquoi les entrer dans le Batch ?
Vous verrez ultérieurement que selon qu'une condition soit vraie ou fausse, la ligne « `Goto Jmp1` » pourra être exécutée ou non. Dans ce cas, l'exécution des commandes 3, 4, et 5 sera *conditionnelle* – vous pourrez donc exécuter certaines instructions dans des cas précis uniquement.

Dernières précisions sur les commandes de saut :

- Deux labels ne peuvent pas porter le même nom,

- Un « goto » pointant sir un label inexistant génère une erreur lors de l'exécution.

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 : seule la fin de la ligne est exécutée, par conséquent seule une unique 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 :

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

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 conduit 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 a 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 exécutée.
C'est en fait "l'inverse" de la commande IF.

Exemple :

```
If not "%ScoreJoueur%"=="%ScoreMax%" echo Peux mieux faire !
```

IF EXIST *Fichier*

Vérifie l'existence du fichier désigné. S'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. *NOTE : Le travail avec ERRORLEVEL est approfondi ultérieurement.*

ATTENTION. Si vous devez tester plusieurs valeurs 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 résultat de la question.

Voilà comment nous allons procéder :

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

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

```
@echo off
If not "%1"=="/?" Goto Suite
Echo Voici l'aide de ce programme
Echo Bla bla bla bla
Goto fin
:Suite
Echo Pour commencer, pressez une touche
Pause
:Fin
REM Fin du batch
```

Dans le cas ci-dessus, si le paramètre envoyé au batch n'est pas "/?"; les commandes après "Suite" sont exécutées. Sinon, le texte d'aide est affiché. On a utilisé un second label « FIN » afin que dans le cas ou le texte d'aide est affiché, le programme ne soit pas ensuite exécuté, mais que celui-ci se termine au contraire directement.

Les capacités de la commande IF sont grandement supérieures sous Windows NT (Voir annexe II)

6°) Boucles avec FOR

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.

Les capacités de la commande IF sont grandement supérieures sous Windows NT (Voir annexe II)

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.

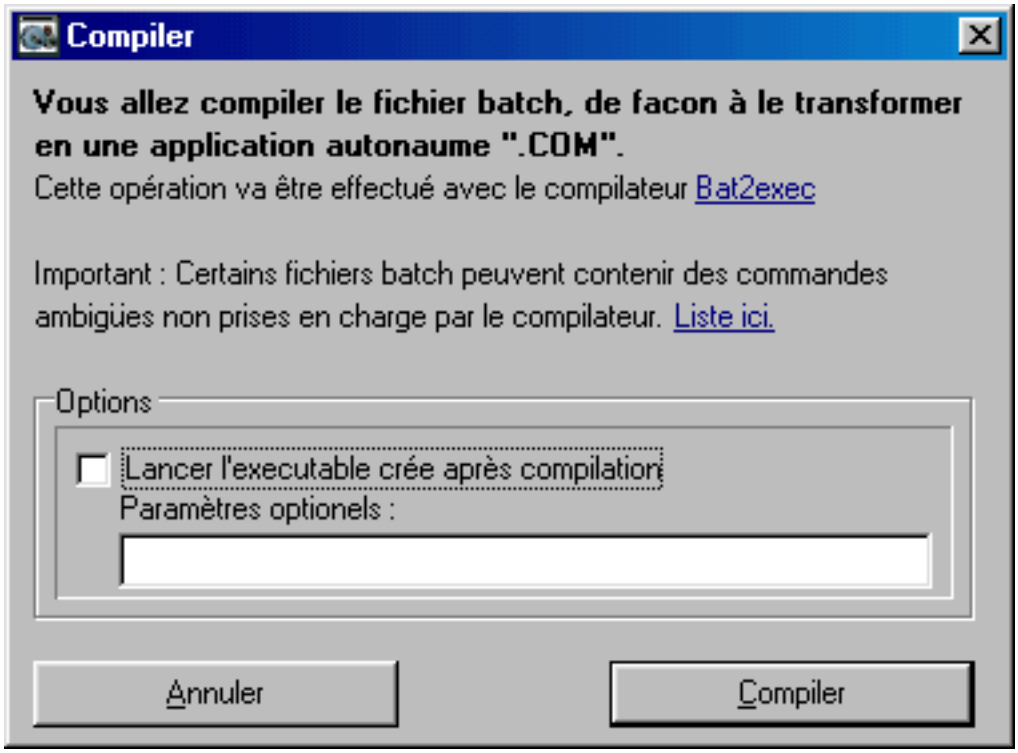
Compiler un fichier

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.

La fenêtre suivante 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 une boite de dialogue confirmant le succès de la compilation.

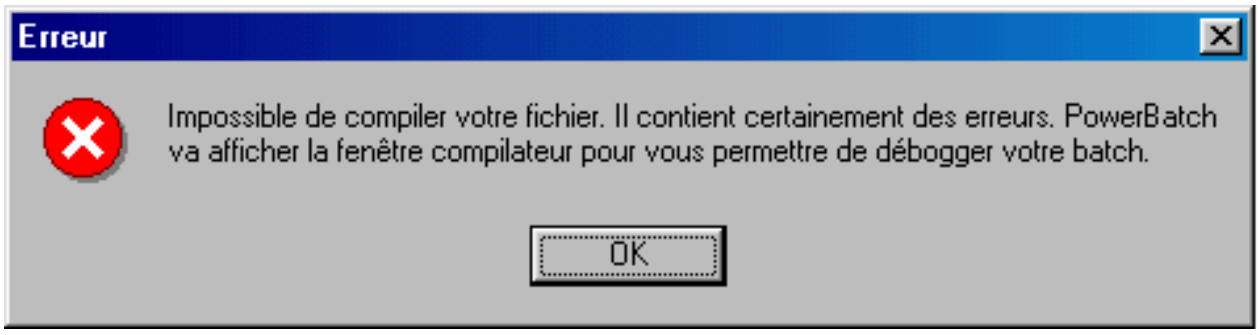
Compilation avec erreur

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

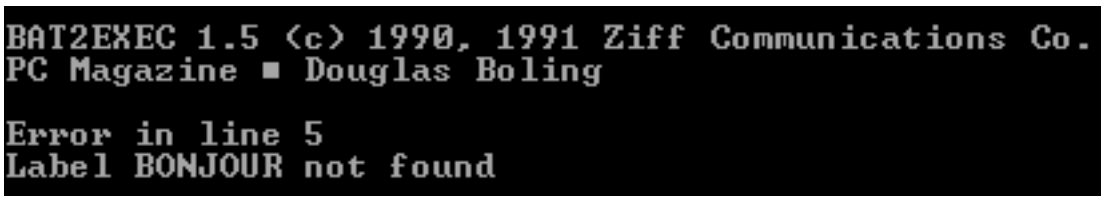
Si par exemple, nous introduisons une erreur dans notre code…

```
@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 :



Bat2exec va vous montrer l'erreur qu'il a rencontrée, dans notre cas, on a :



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'où vient l'erreur d'après le n° de ligne transmis par Bat2exec.

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 façon :

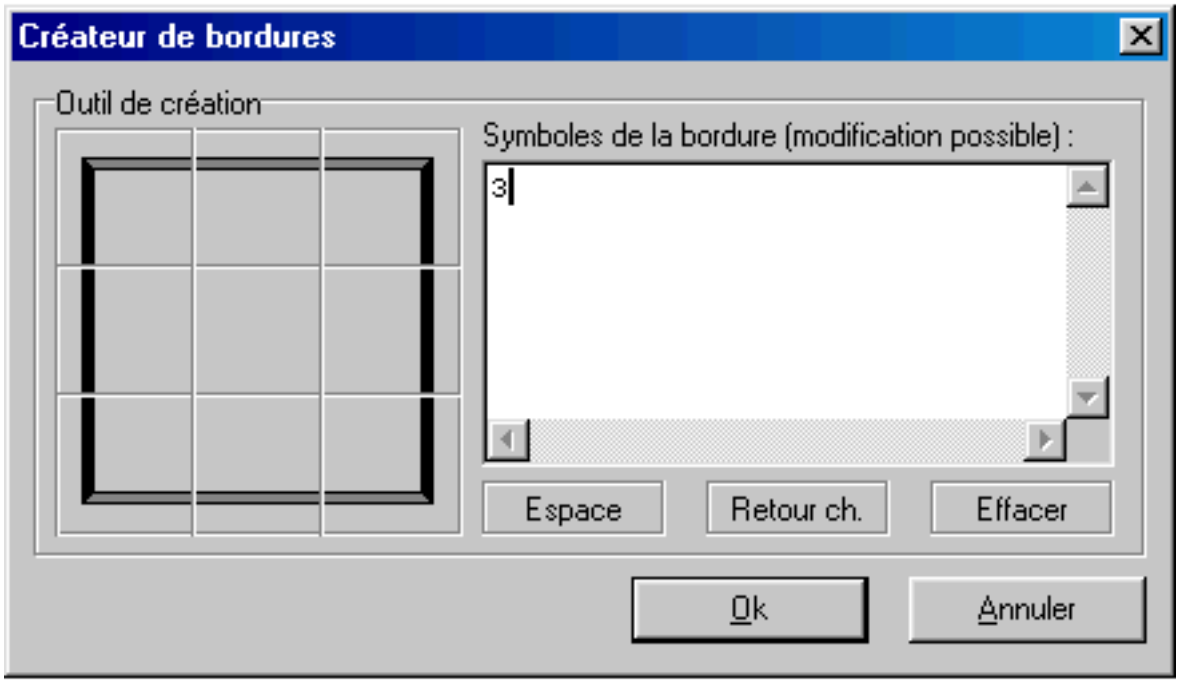
Pour cela, MS-DOS utilise tous les caractères "spéciaux", c'est pour cela que dans le chapitre 1 nous vous avons conseillé d'éviter d'utiliser les caractères accentués tels que "é,ç,à" etc…

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 :

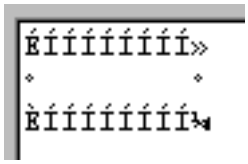
et 1 fois sur la case :

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 :

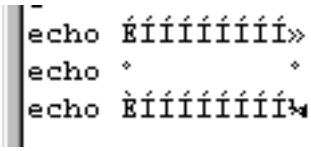


Ne nous préoccupons pas pour l'instant du texte "Bonjour" à intercaler dans la bordure.
Notre bordure à proprement parler est maintenant créée. Pour l'insérer dans le fichier Batch, cliquez sur :

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".

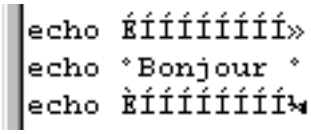
Dans votre Batch, vous avez maintenant :



```
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°) Redirections de sorties
et écriture dans les fichiers

Écrire 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 dans ce chapitre on va voir comment 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 pas eux-mêmes,
• 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 le programme 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 le ">" 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-mêmes 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-mêmes pas. Par conséquent, si les dossiers "MonJeu" et "Scores" ne sont pas présents 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 ou **Set CPT=**%1 par exemple.

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.
La commande à gauche de « >nul » est exécutée correctement, mais son résultat n'est pas affiché.

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 exist g:\NUL faitquelquechose` , "if" testant si un fichier virtuel pouvant représenter n'importe quel élément en réalité sur le disque existe.

La redirection vers "prn" et les autres ports

Évoquons brièvement la redirection vers « Prn » : elle permet d'envoyer du texte à l'imprimante :

```
Echo test>prn
```

De la même façon, vous pouvez utiliser à la place de *prn* tout autre symbole désignant un port : *lpt1*, *lpt2*, *com1*, *com2*… mais cela dépasse le cadre du tutoriel.

Le symbole de redirection « | »

Ce caractère de redirection, appelé « Pipe » (canal, tuyau) en anglais permet de rediriger la sortie d'une commande vers l'entrée d'une autre. Ce caractère s'obtient en maintenant la touche ALT enfoncé en tapant 124 au clavier (ou AltGr + 6).

Prenons un exemple bien connu : en tapant « DEL *.* » afin d'effacer le contenu d'un répertoire, un message de confirmation apparaît, demandant l'appui sur la touche « o » (pour OUI) pour continuer. Il serait intéressant de supprimer ce message de confirmation qui interrompt le déroulement d'un batch – afin que la commande s'exécute « toute seule » sans besoin d'une validation quelconque de l'utilisateur. La commande suivante permet de résoudre ce problème :

```
Echo o|del*.*
```

La « sortie » de la commande *echo* (le caractère « o ») est envoyée vers « l'entrée » de la commande DEL. Lorsque le message sera affiché, MS-DOS considèrera le caractère reçu comme une réponse au message et continuera l'exécution du batch. Dans notre cas, ce sera comme une « validation » à la question « Pressez o pour continuer, n pour annuler ».

Second exemple : A votre avis que fait la commande suivante ?

```
ECHO.|DATE >> sauv.lst
```

Réponse : La date est sauvegardée dans le fichier SAUV.LST – Mais le message « Entrez une nouvelle date (jj.mm.aa) : » est validé par un retour chariot « echo . » : la sauvegarde se fait sans besoin d'intervention humaine (pas besoin de valider lorsque le message demandant d'entrer une nouvelle date apparaît).

Ce symbole facilite grandement le vie du programmeur Batch, qui peut ainsi imposer « d'avance » la réponse à des messages DOS qui pourraient entraver l'exécution de ses batchs.

Note : On retrouve aussi ce caractère dans l'utilisation quotidienne du DOS : le fait d'accoler « `|more` » à une commande standard permet que son résultat, s'il prend plusieurs écrans, ne soit pas affiché « d'un coup » mais en autant de fois qu'il le faut pour faciliter la lecture. Ainsi, testez sous DOS :

```
CD C:\WINDOWS
DIR *.*|MORE
```

(Bien sûr remplacez « C:\WINDOWS » par l'adresse du dossier de Windows chez vous si ce n'est pas la même). La liste sera affichée en plusieurs fois : à chaque écran, le DOS vous demande de presser une touche pour continuer le défilement.

Rediriger une entrée vers une commande : le caractère « < »

Beaucoup de commandes nécessitent l'envoi de caractère pour valider des messages ou répondre à des questions. MS-DOS vous permet de répondre à ces questions en lisant le texte à envoyer dans des fichiers.

Par exemple : créez avec EDIT ou NOTEPAD un fichier à la racine de C:\ nommé test.txt. Inscrivez « 6/12/02 », puis sauvez votre fichier.
Ouvrez ensuite une console DOS, placez vous à la racine de C:\ si ce n'est pas déjà le cas, puis entrez :

```
DATE<TEST.TXT
```

Quelques secondes plus tard, la date actuelle aura été changé en 6/12/02 !

Que s'est-il passé ? Le caractère « < » (chevron orienté dans le sens INVERSE que pour l'écriture) à redirigé le contenu du fichier TEST.TXT dans « DATE ». Ainsi, lorsque DATE a « demandé » une nouvelle date, c'est le contenu du fichier qui a validé la commande. Là aussi, vous avez réussi à supprimer toute intervention humaine pour entrer et valider une nouvelle date.

Question : aurait-on pu imposer une réponse sans lire dans un fichier avec l'aide d'une autre commande ?
Bien sûr que oui, en utilisant le « Pipe » | de la facon suivante :

```
ECHO 6/12/02|DATE
```

…Mais l'utilisation du chevron de *lecture* « < » permet de définir une date *suivant le contenu* du fichier : celle-ci n'est pas fixée une fois pour toute dans le batch comme avec la commande de Pipe, mais peut varier selon le contenu du fichier texte.

Un autre exemple :

• Pris au hasard dans un titre de la presse informatique (attention : ce n'est qu'un extrait, cet exemple n'est pas complet !):

```
echo E 01D0 03 50 A4 92 F2 BF 4C 25 3E 67 86>>scr2bin
echo RCX>>scr2bin
echo 1db>>scr2bin
echo W 0>>scr2bin
debug < scr2bin > nul
```

Les quatre premières lignes écrivent des données dans un fichier « scr2bin ». Celui-ci est écrit en mode « Ajout » en utilisant les doubles chevrons – chaque ligne étant inscrite les unes après les autres.
La cinquième ligne est plus complexe : on peut d'abord voir que le résultat de cette ligne ne sera pas affiché : on utilise la redirection vers *nul* via « `> nul` », et que la commande « *debug* » est utilisée. Au milieu, il reste « `< scr2bin` » : on comprend donc que le fichier « scr2bin » est envoyé au programme « debug » afin d'être interprété. Le fichier est donc crée par le batch, puis envoyé à *debug*, en masquant les commentaires/erreurs pouvant –être affichées par le programme.

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 l'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****. Si vous programmez avec Delphi vous pourrez utiliser* ***Halt(x),*** *où x désigne la valeur de retour à renvoyer au DOS.*

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
```

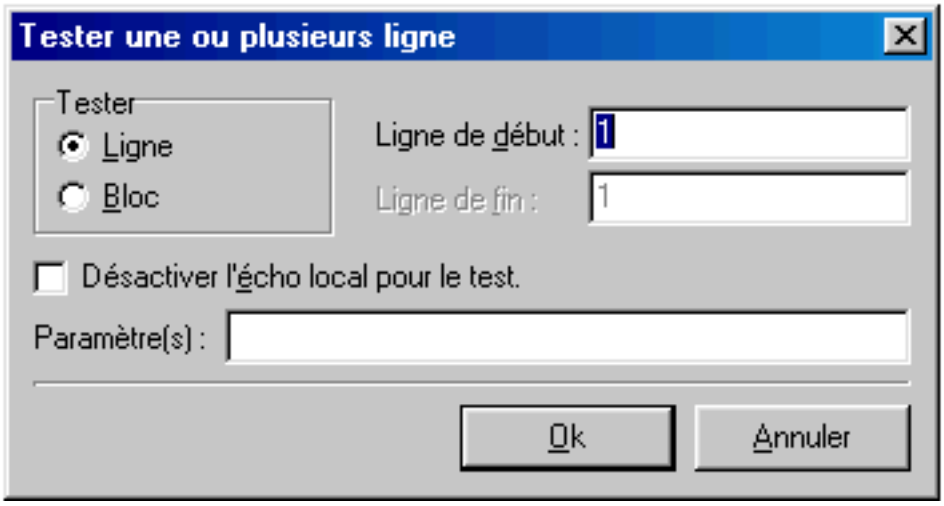
Toutes les commandes DOS ne renvoient pas des valeurs d'erreur. Les commandes concernées n'utilisent que certaines valeurs.

12°) 5 autres fonctions de PowerBatch

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

PowerBatch présente diverses possibilités de test de vos fichiers batch :

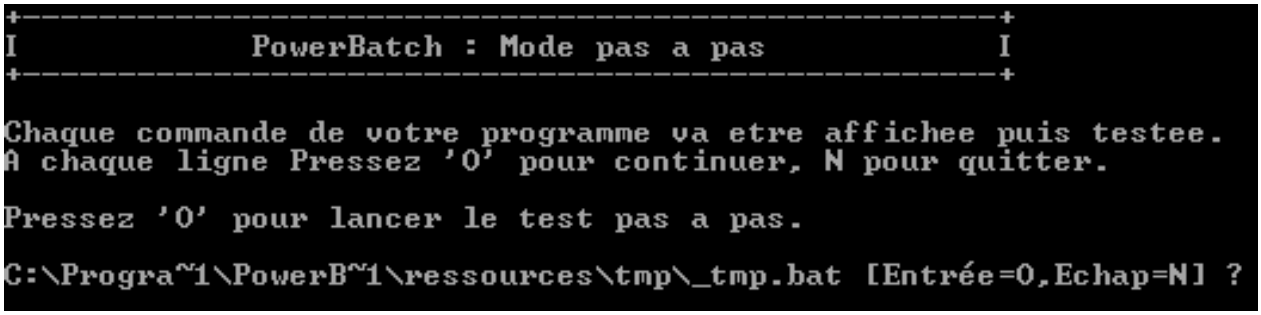
• Le test ligne/bloc est obtenu en pressant la touche F8 (ou avec le menu Programme>Deboggage) :



Cette fonction vous permet de tester une seule ligne de votre fichier ou un bloc de commandes, de la ligne X à la ligne Y.

Vous pouvez passer des paramètres au batch ou désactiver l'écho local (echo off) pour vos commandes.

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 » d'exécuter la ligne, « N » de la passer sans l'exécuter.
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 plusieurs modèles complets de Batch.
Le plus utile est sans doute "Environnement étendu" car il permet de parer les erreurs dues à un espace d'environnement insuffisant.
De plus, y sont ajoutés deux "macros", ensembles 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 pressée, 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.

Avec CHOICE, il devient très facile de créer des petits menus à choix uniques.
ATTENTION : Cette commande n'est pas incluse sous Windows NT !
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 possédant 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 met également en relief le code converti (les commandes DOS seront distinguées, les commentaires mis en italique etc.).

Le fichier sera crée, et une nouvelle icône apparaîtra à l'endroit désigné. Double-cliquez sur l'icône pour lancer le navigateur associé aux fichiers HTML (en général Microsoft Internet Explorer)

13°) Programmation avancée

Dans ce bref et dernier chapitre, mon but est de vous montrer que l'on peut vraiment tout faire avec le langage batch, si l'on a en quelque sorte une âme de « bricoleur ».

J'ai dit dans le premier chapitre que des fonctions comme le traitement numérique, l'entrée clavier (de mots, et non pas de lettres, telles qu'on le fait avec la commande CHOICE) n'était possible qu'avec l'aide de programme externe.

En fait, au moyen de plusieurs astuces, on peut réaliser certaines fonctions intéressantes :

- Par le détournement de certaines commandes (utilisation de certaines de leurs fonctions pour réaliser d'autres tâches que celles prévues à l'origine),

- Par le « Scriptage » de certains logiciels : en effet, certains programmes peuvent exécuter des commandes inscrites dans des fichiers (ex : debug ou ftp), il suffit donc d'écrire dans des documents les instructions à exécuter par le programme puis de l'appeler en lui indiquant quel fichier de commandes exécuter,

- Par la redirection des résultats de commande dans des fichiers, puis la recherche et l'extraction d'informations à partir de commandes spécifiques (commande FIND par exemple)

Exemple : ce fichier Batch détermine si Windows est lancé, si l'on est sous DOS ou si nous sommes en mode DOS réel, juste après avoir quitté Windows :

Ce listing fait appel au programme MEM (affichage de la mémoire libre/utilisée sur le système), et tente de chercher certains fichiers en mémoire significatifs de l'utilisation de MS-DOS. Suivant les résultats, il affiche un message correspondant.

```
@echo off
cls
mem /m win | find "K"
if errorlevel 1 goto never
mem /m vmm32 | find "K"
cls
if errorlevel 1 goto dosmode
goto doswin
:never
echo La machine n'a pas encore demarre sous Windows.
goto done
:doswin
echo Windows est actuellement lance.
goto done
:dosmode
echo Windows a deja ete lance mais vous avez redemarre en mode
MS-DOS
:done
```

Vous pouvez faire mieux ! Le programme DEBUG, permettant entre autres d'afficher l'état de la mémoire, peut être utilisé pour calculer la taille de mémoire disponible sur un disque, etc… Néanmoins, ces solutions s'avèrent peu pratiques, lourdes à mettre en œuvre et plutôt lentes.

Si vous souhaitez vous lancer dans la « haute voltige » de la programmation Batch, jetez un coup d'œil sur ces pages:

- http://home7.inet.tele.dk/batfiles/
- http://www.fpschultze.de/batstuff.html
- http://www.cableyorkton.com/users/gbraun/batch/
- http://www.merlyn.demon.co.uk/batfiles.htm
- http://www.cybertrails.com/~fys/index.htm

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.com

Si vous remarquez des erreurs, ou pensez que des compléments sont nécessaires, merci de me contacter via le site web !

Bonne continuation…

Adrien REBOISSON

ANNEXE 1] Intégration MS-DOS dans Windows 9x

Source : PC Poche Pratique – Optimisation des fichiers système de Windows ® 95 – Micro Application (ISBN : 2-7429-0659-2)

1°) Caractéristiques techniques pour Windows 9x

6) Si vous ouvrez une fenêtre DOS sous Windows, MS-DOS 7.0 est lancé, les fichiers CONFIG.SYS et AUTOEXEC.BAT sont utilisés pour installer un environnement de travail pour tous les programmes,

7) Si un programme refuse de fonctionner dans la fenêtre DOS ou en plein écran, vous devrez le lancer en *mode MS-DOS*, un pur mode réel du DOS suffisamment identique à l'ancien MS-DOS pour que tous les programmes DOS puissent fonctionner dans ce mode.

8) MS-DOS fonctionne maintenant en « Machine virtuelle », presque exclusivement dans le mode protégé du processeur. Un petit nombre de pilotes en mode réel, une espèce en voie de disparition, sont exploités dans le mode 8086 virtuel. Le nouveau DOS est plus rapide et plus stable,

9) Ces machines virtuelles peuvent être ouvertes en nombre pratiquement illimité, l'utilisation simultanée de plusieurs programmes DOS est donc possible sans causer de problèmes,

10) Chaque machine virtuelle peut-être configurée indépendamment des fichiers de configuration comme AUTOEXEC.BAT et CONFIG.SYS,

11) Les noms de fichiers longs peuvent êtres utilisés dans une fenêtre DOS. Il faut simplement se souvenir que l'espace ne fonctionne plus comme séparateur entre les paramètres, mais qu'il faut utiliser des guillemets :

```
COPY «Exemple de texte.doc» C:\TEXTES\CHP1
```

12) La taille des fenêtres DOS est modifiable car les caractères sont affichés en des polices « TrueType ». La taille des caractères est automatiquement adaptée à celle de la fenêtre dans le mode AUTO,

13) Les opérations de couper/copier/coller dans des fenêtres DOS sont gérées, et les presse-papier peut-être utilisé entre les applications DOS Windows :

14) Le chargement des pilotes souris devient inutile, car la souris est automatiquement gérée en mode fenêtre ou plein écran,

15) Vous avez plus de mémoire : presque tous les pilotes importants comme NET.EXE ou SHARE.EXE existent en version 32 Bits et peuvent utiliser toute la mémoire, y compris celle située au-delà de la limite des 1 Mo. Les programmes qui refusaient de fonctionner pour cause de mémoire insuffisante ont une nouvelle chance…

16) Tous les paramètres d'une fenêtre DOS peuvent être entrés dans une boîte de dialogue « Propriétés ». L'éditeur PIF appartient au passé ; les données sont toujours stockées dans les fichiers PIF pour des raisons de compatibilité mais cela ne vous concerne pas (en fait les données sont enregistrées dans la base des registres, et le fichier PIF se contente de pointer vers elles)

17) La convention de noms UNC est utilisée dans la fenêtre DOS :

```
DIR \\TEST\FILES\TEXTS
```

... Permet un accès direct à une ressource partagée par la création dans le réseau d'un nom similaire précédé de :

```
\\NOMSERVEUR\NOMPARTAGE
```

2°) Préférences d'Exécution et fichiers PIF

Les fichiers PIF, contenant les paramètres d'exécution pour les programmes DOS lancés sous Windows, existent encore sous Windows 95, mais l'éditeur si chétif a été remplacé par une boîte de dialogue de configuration bardée d'onglets.
L'utilisateur n'a plus besoin d'enregistrer les données explicitement comme fichier PIF. Quand il a terminé sa saisie, le fichier PIF est automatiquement enregistré, sous même nom que le programme et dans le même répertoire que celui-ci.
L'utilisateur n'a plus qu'à cliquer sur le fichier PIF plutôt que de lancer directement le programme pour que celui-ci soit démarré avec les préférences d'exécution choisies.

Ci-dessus : L'exécutable PB.EXE et son fichier PIF associé PB.PIF

Pour éditer les préférences d'exécution du programme, vous devez cliquer sur le bouton droit de la souris, choisir la commande « Propriétés », et choisir les options proposées dans les divers onglets de la boîte de dialogue.

Les préférences du programme

L'onglet ***Programme*** est réservé à la modification des paramètres de base, comme le nom du programme ou le répertoire utilisé.

Nom : Le nom à afficher dans la barre de titre du programme.

Ligne de commande : Entrez ici le chemin d'accès ainsi que les paramètres éventuellement nécessaires à l'appel du programme.

Répertoire de travail : C'est le répertoire utilisé par le programme pour lire et écrire ses données (textes dans le cas d'un traitement de texte)

Fichier de commandes : Entrez ici le nom du fichier batch que vous désirez exécuter avant le lancement du programme. Ce fichier peut par exemple déclarer et initialiser des variables d'environnement ou effectuer des copies de secours de certains fichiers. Vous pouvez également entrer l'adresse d'un autre programme EXE ou COM ou une commande dos à exécuter.

Touche d'accès rapide : Vous pouvez affecter dans cette zone certains raccourcis clavier pour lancer directement l'application.

Exécuter : Comment afficher la fenêtre (normale, en plein écran, etc…)

Fermer en quittant : Désirez vous que la fenêtre DOS soit automatiquement fermée une fois l'application terminée ?

Le bouton ***Changer d'icône*** permet d'affecter une autre icône au programme.

Le bouton ***Paramètres avancés*** permet d'aller plus loin :

• L'option ***Empêcher la détection de Windows par des programmes MS-DOS*** permet de forcer le fonctionnement des applications DOS qui refusent de tourner si elles détectent la présence de Windows. Cette option est inactivée par défaut. Si elle est active, le programme ainsi configuré ne recherche pas la présence de Windows.

• L'option ***Suggérer le mode MS-DOS approprié*** signifie que Windows vérifie qu'il peut exécuter le programme correctement. S'il s'agit d'un programme « spécial », le mode MS-DOS est suggéré et configuré dans la boîte de dialogue. Si vous inactivez cette option, un programme « délicat » sera lancé malgré les risques de plantage.

Les polices de caractère

L'onglet ***Police*** permet d'affecter une police de caractère au programme. Ce paramètre n'est utile qu'avec les programmes fonctionnant en mode texte en non en mode graphique.

La partie gauche indique si la police existe en BitMap, en TrueType ou dans les deux formes. Vous pouvez également définir la taille des caractères. Les deux zones du bas reproduisent l'effet du paramètre sélectionné sur la taille de la fenêtre (à gauche) et sur celle des caractères (à droite).

Avec la taille AUTO, la taille des caractères sera automatiquement adaptée à celle de la fenêtre.

L'allocation de la mémoire

L'allocation des divers types de mémoire est presque entièrement automatique sous Windows 9x installé sur un ordinateur moderne.
Vos interventions se limiteront aux cas les plus complexes.

Vous pouvez entrer la quantité de mémoire conventionnelle, EMS (étendue), et XMS (paginée) nécessaire au bon fonctionnement de cette application, ainsi que la mémoire à conserver en mode protégé.

Si vous cochez l'option ***Protégé*** de la mémoire conventionnelle, votre programme sera protégé contre les accès « sauvages » de ce programme à la mémoire. Ce mode de fonctionnement met en œuvre des procédures de vérification activées à intervalles très rapprochés, ce qui ralentit la vitesse d'exécution du programme, mais cela vous assure une meilleure stabilité du système.

Gérer la fenêtre

Les paramètres de cet onglet concernent l'affichage, à l'exception des deux dernières options (grouper ***Performance***), dont la place est sur l'onglet précédent.
L'option ***Emulation ROM rapide*** commande au pilote graphique la simulation des fonctions vidéo de la mémoire ROM, ce qui apporte un gain en performance.
La deuxième option, ***Allocation de la mémoire dynamique*** permet à Windows d'utiliser immédiatement la mémoire libérée par le passage à une application tournant en mode texte, ce qui occupe moins de mémoire que les applications en mode graphique.

Les autres options de cet onglet concernent l'aspect de la fenêtre : affichage en mode fenêtre ou plein écran, affichage de la barre d'outils et utilisation des paramètres Taille, Position et Police de la fenêtre doivent être utilisés à chaque démarrage de ce programme.

L'onglet « Divers »

Le nom de cet onglet peut faire croire qu'il ne contient que quelques options d'importance secondaire, or c'est exactement l'inverse.

Les paramètres accessibles dans l'onglet ***Divers*** peuvent par exemple, s'ils sont maladroitement définis ; vous interdire de quitter une fenêtre DOS.

Explication des options des options mises à votre disposition :

Autoriser l'écran de veille : L'activation de cette option inclut le programme en cours d'exécution dans la fenêtre DOS dans la surveillance de l'inactivité de l'utilisateur par l'économiseur d'écran Windows.

Édition rapide : Cette option permet de sélectionner des éléments de la fenêtre DOS en vue de l'utilisation du presse-papiers. Dans la configuration par défaut, vous devez utiliser le menu.

Mode exclusif : Si la souris cause des problèmes, ce mode peut y remédier tout en autorisant l'usage de la souris. Attention tout de même : la souris n'est plus utilisable sous Windows 95, et vous ne pourrez plus revenir à l'interface graphique en cliquant au-dehors de la fenêtre DOS.

Toujours suspendre : Quand cette option est activée, l'exécution du programme est suspendue lorsqu'il n'est pas utilisé.

Avertir si encore actif : Si cette option est activée, un message apparaît lorsque vous tentez de fermer un programme DOS encore actif.

Sensibilité à l'attente : Ce curseur affecte au programme un délai pendant lequel le processeur attend votre reprise d'activité avant d'affecter son temps de calcul à une autre application. Pour augmenter les ressources du programme, déplacez le curseur en direction de ***Basse,*** pour diminuer ses ressources, rapprochez le curseur de ***Haute.***

Collage rapide : Cochez cette case pour coller les données du presse-papier en mode rapide. Si les données ne sont pas correctement collées, alors décochez cette case.

Touches de raccourci de Windows : Si certains raccourcis normalement réservés à Windows sont importants dans votre application DOS, vous pouvez désactiver la prise en charge des raccourcis Windows décochés dans cette section.

3°) Fonctionnement en mode MS-DOS

Lorsque Windows est redémarré en *mode MS-DOS,* il se réduit à un petit module de démarrage dans la mémoire et laisse un environnement de travail adéquat pour le mode MS-DOS en mode réel :

- Le PC tourne en mode réel calqué sur celui de la version 6.0 de MS-DOS,

- Tous les programmes Windows sont arrêtés,

- Les fonctions de réseau ne sont plus disponibles, mais elles peuvent être chargées par la commande NET du fichier AUTOEXEC.BAT ou dans la configuration individuelle de démarrage.

Windows redémarre automatiquement quand vous quittez l'application MS-DOS.
Toutes les applications doivent avoir été terminées correctement et leurs données enregistrées.

Quand vous configurez un programme pour le mode MS-DOS, via l'onglet ***Paramètre de programme avancés*** de nouvelles options s'ouvrent à vous.

Pour que le programme soit démarré en mode MS-DOS réel, vous devez cocher la case ***Mode MS-DOS.***

Si vous désactivez l'option ***Avertir avant de passer en mode MS-DOS*** vous ne serez pas invité à refermer toutes les autres applications avant le passage de Windows dans ce mode.
Ne décochez cette case que si vous avez pris l'habitude de fermer les applications et d'enregistrer vos données avant de lancer le programme.

Ensuite vous avez le choix entre :

- Utiliser la configuration actuelle, définie entre autres par AUTOEXEC.BAT et CONFIG.SYS pour le programme,

- Utiliser une configuration spécifique de AUTOEXEC.BAT et CONFIG.SYS pour le programme. Dans ce cas, les zones de texte « CONFIG.SYS » et « AUTOEXEC.BAT » doivent être remplies avec le contenu des fichiers adéquat comme s'ils étaient réellement lancés depuis votre disque.

Si vous souhaitez utiliser une configuration spécifique sans entrer de lignes de code, cliquez sur le bouton « Configuration ».

Dans la boîte de dialogue, cochez les modules dont vous avez besoin et laissez à Windows le soin d'inscrire la ligne adéquate dans le liste de configuration.

Enfin, signalons également que s'il existe un fichier ***DOSSTART.BAT*** sur votre système, celui-ci sera exécuté lors du lancement du mode réel. Vous pourrez donc inscrire toutes les commandes que vous désirez utiliser dans tout appel au mode MS-DOS.

Pensez toujours à tester des programmes DOS récalcitrants sous Windows en mode DOS réel. Dans la majorité des cas, vos problèmes seront résolus.

Information : Les préférences d'exécution standard pour toutes les fenêtres DOS sont définies dans le fichier COMMAND.PIF. Si vous modifiez les propriétés dans la fenêtre DOS, chaque programme DOS sera lancé sur la base des paramètres modifiés.

ANNEXE 2] Différences de l'intégration Batch entre Windows et Windows NT

Windows NT (terme regroupant tout OS basé sur un noyau NT 32 bits : Windows NT4, Windows 2000, et Windows XP) possède un « moteur » d'intégration Batch tout a fait différent que les éditions « personnelles » de Windows. MS-DOS a presque disparu du système (néanmoins compatible avec les programmes écrits pour cet OS), et la traditionnelle « Console MS-DOS » s'est transformée en « Invité de commande » prenant en charge la majorité des anciennes commandes DOS, mais ajoutant aussi beaucoup d'extensions : Windows NT conserve et améliore presque toutes les fonctionnalités de MS-DOS.

Ce chapitre propose un aperçu des différences majeures entre Windows XP (Windows XP étant un système NT, les informations relatives à cet OS peuvent s'appliquer à Windows 2000 et dans une moindre mesure à Windows NT4) et les éditions personnelles de Windows (Windows 95, Windows 98 et Windows Me). Pour plus d'informations, tapez « HELP » dans l'invite de commande spécifique, ou entrez une commande, suivie de « / ? ».

• Les commandes ajoutées dans Windows XP sont les suivantes : `at, cacls, convert, diskperf, dosonly, echoconfig, endlocal, findstr, ntcmdprompt, popd, pushd, pushd, setlocal, start, title, &&,||,&,( ),.,^,;,.`

• Les commandes modifiées dans Windows XP sont les suivantes : `chcp, cmd, del , dir, diskcomp, diskcopy, doskey, format, vkeyb,label, mode, more, path, print, prompt, recover, rmdir, sort, xcopy.`

• Les commandes non prises en charge par Windows XP par rapport à Windows 95/98 sont les suivantes : `assign, choice, ctty, dblspace, defrag, deltree, dosshell, drvspace, emm386, fasthelp, fdisk, include, interlnk, intersrv, join , memmaker, menucolor, menudefault, menuitem, mirror, msav, msbackup, mscdex, msd,numlock, power, scandisk,smartdrv, submenu, sys, undelete, unformat, vsafe.`

• Les propriétés, comme le titre de la console ou sa couleur peuvent-être modifiés soit via le menu système d'une console, soit dans vos batchs grâce aux commandes :

`COLOR :` Change les couleurs par défaut du premier et de l'arrière plan de la console (pour plus d'informations, tapez dans une console « `COLOR / ?` »

`TITLE :` Définit le titre de la fenêtre pour une fenêtre DOS

Par exemple :

```
Color 10
Title Copie des fichiers
```

… Modifie le titre de console en « Copie des fichiers », et affiche du texte noir sur un fond bleu-foncé.

• Les extensions des scripts console sont « .bat », mais aussi « .cmd »

• L'*historique des commandes* permet de relancer une commande déjà tapée. L'historique complet est obtenu par la touche F7, la touche F8 permet de rechercher dans l'historique la commande la plus récente ayant le même départ que ce qui est indiqué sur la ligne de commande.

• Grande amélioration de la commande IF :

- Possibilité d'inclure la clause « ELSE » qui introduit du code exécuté si la condition n'est pas satisfaite. « ELSE » doit être sur la même ligne que la commande suivant IF. Vous pourrez ainsi écrire :

```
IF EXIST NomFichier (
     DEL NomFichier
) ELSE (
     ECHO NomFichier est introuvable !
)
```

…ou bien :

```
IF EXIST NomFichier (DEL NomFichier) ELSE ECHO NomFichier
est introubale
```

Dans les deux cas, si NomFichier n'existe pas, le texte d'erreur est affiché, sinon, le fichier est effacé.

Si les extensions de commande sont activées, les commandes ci-dessous sont utilisables :

• `IF Chaine1 OP Chaine2 Commande`

OP doit-être remplacé par :
EQU : égal à,`
NEQ : différent de,
LSS : inférieur à (<),
LEQ : inférieur ou égal à,
GTR : supérieur à (>),
GEQ : supérieur ou égal à.

Ainsi, la commande ci-dessous détecte une exécution correcte d'un programme en allant à OK si ERRORLEVEL est inférieur ou égal à 1:

```
IF %ERRORLEVEL% LEQ 1 Goto OK
`
```

- `IF DEFINED Variable Commande`

  Cette commande est identique à « IF EXIST », mais fonctionne avec des variables. Ainsi, si celle-ci est définie (existe, a été déclarée ultérieurement), la commande « Commande » est exécutée.

- `IF CMDEXTVERSION nombre Commande`

  …ou %CMDEXITVERSION% est une chaîne représentant la valeur actuelle de la version de l'interpréteur de commandes. Ce numéro sera incrémenté à chaque amélioration significative de l'interpréteur. Si des nouveautés apparaissent dans une version future de Windows, vos batchs pourront détecter la version de l'interpréteur sur laquelle ils fonctionnent et exécuter des commandes en conséquence.

• Nouvelle commande « SET» *(Commentaires extraits de l'aide en ligne sur la commande)*

Deux nouvelles options ont été ajoutées à la commande SET :

```
SET /A expression
SET /P variable=[ChaîneInvite]
```

L'option /A spécifie que la chaîne à droite du signe égal est une expression numérique qui est évaluée. L'évaluation de l'expression est assez simple et prend en charge les opérations suivantes dans l'ordre décroissant de préséance :

```
()                    - groupement
! ~ -                 - opérateurs unaires
* / %                - opérateurs arithmétiques
+ -                   - opérateurs arithmétiques
<< >>                 - décalage logique
&                     - ET au niveau du bit
^                     - OU exclusif au niveau du bit
|                     - OU au niveau du bit
= *= /= %= += -=     - attribution
  &= ^= |= <<= >>=
,                     - séparateur d'expression
```

Si vous utilisez des opérateurs logiques ou des nombres, vous devez mettre l'expression entre guillemets. Toute chaîne non numérique dans l'expression est traitée comme une variable d'environnement dont les valeurs sont converties en nombres avant d'être utilisée. Si un nom de variable d'environnement est spécifié mais n'est pas défini dans l'environnement en cours, alors la valeur zéro est utilisée.

Cela vous permet de faire des opérations avec les valeurs d'une variable d'environnement sans avoir à entrer des signes % pour obtenir ces valeurs. Si SET /A est exécuté à partir de la ligne de commande en dehors d'un script de commande, alors la valeur finale de l'expression est affichée.

L'opérateur d'assignation requiert un nom de variable d'environnement à gauche du cet opérateur. Les valeurs numériques sont des nombres décimaux, à moins qu'ils ne soient préfixés par 0x pour les valeurs hexadécimales et 0 pour la notation octale. Donc 0x12 est identique à 18 et à 022. Notez que la notation octale peut être confuse : 08 et 09 ne sont pas valides car 8 et 9 ne sont pas des nombres valides en notation octale.

L'option /P vous permet de fixer la valeur d'une variable avec une ligne entrée par l'utilisateur. Elle affiche la chaîne ChaîneInvite spécifiée avant de lire la ligne entrée. La ChaîneInvite peut être vide.

La substitution de la variable d'environnement a été améliorée comme suit :

```
%PATH:ch1=ch2%
```

…développe la variable d'environnement PATH, remplaçant chaque occurrence de "ch1" dans le résultat développé par "ch2". "ch2" peut être une chaîne vide pour supprimer toutes les occurrences de "ch1" de la sortie développée. "ch1" peut commencer par un astérisque, auquel cas la commande traitera la chaîne à partir du début et jusqu'à la première occurrence du reste de ch1.

Vous pouvez aussi spécifier des sous-chaînes pour une expression :

```
%PATH:~10,5%
```

…développe la variable d'environnement PATH et utilise seulement 5 caractères à partir du onzième (décalage de 10) dans le résultat sous forme développée. Si la longueur n'est pas spécifiée, tout le reste de la valeur de la variable est traité. Si l'une des valeurs (décalage ou longueur) est négative, alors le nombre utilisé est la longueur de la valeur de la variable d'environnement ajoutée au décalage ou à la longueur donné.

```
%PATH:~-10%
```

…extrait les 10 derniers caractères de la variable PATH.

```
%PATH:~0,-2%
```

…extrait tous les caractères sauf les deux derniers de la variable PATH.

Enfin, la prise en charge de l'expansion retardée de variables d'environnement a été ajoutée. Cette prise en charge est toujours désactivée par défaut, mais peut être activée/désactivée via l'option de ligne de commande /V dans CMD.EXE. (Voir CMD /?)

La prise en charge de l'expansion retardée de variables d'environnement est utile pour contourner les limites de l'expansion en cours qui se produit à la lecture d'une ligne de texte et non à son exécution. L'exemple suivant montre le problème causé par l'expansion immédiate de variables :

```
    set VAR=avant
    if "%VAR%" == "avant" (
        set VAR=après
        if "%VAR%" == "après" @echo Cela marche si vous lisez
ce message
    )
```

…n'affichera jamais le message car la variable %VAR% présente dans les deux instances de IF est remplacée à la lecture de la première instance de IF, étant donné que le corps de IF, instruction composée, est logiquement inclus. Donc IF compris dans l'instance compare "avant" avec "après" et il n'y aura jamais égalité. De même, l'exemple suivant ne fonctionnera pas comme prévu :

```
    set LIST=
    for %i in (*) do set LIST=%LIST% %i
    echo %LIST%
```

…car la liste des fichiers du répertoire ne sera PAS créée, et en fait la variable LIST prendra le valeur du dernier fichier trouvé. Une fois encore cela est dû au fait que %LIST% n'est étendu qu'une fois, lorsque FOR est lu et à ce stade la variable LIST est vide. Donc la boucle FOR exécutée est :

```
    for %i in (*) do set LIST= %i
```

…qui donne toujours à LIST la valeur du dernier fichier trouvé.

L'expansion retardée de variables d'environnement vous permet d'utiliser un autre caractère (le point d'exclamation) afin d'étendre les variables d'environnement durant l'exécution. Si l'expansion retardée de variables est activée, les exemples ci-dessus peuvent être écrits comme suit afin de fonctionner comme vous le souhaitez :

```
    set VAR=avant
    if "%VAR%" == "avant" (
        set VAR=après
        if "!VAR!" == "après" @echo Cela marche si vous lisez
ce message
    )

    set LIST=
    for %i in (*) do set LIST=!LIST! %i
    echo %LIST%
```

Si les extensions de commandes sont activées, alors plusieurs variables d'environnement dynamiques peuvent être développées sans apparaître dans la liste de variables affichée par SET. Les valeurs de ces variables sont calculées dynamiquement chaque fois que la variable est développée. Si l'utilisateur définit explicitement une variable avec un de ces noms, alors cette définition écrase la définition dynamique décrite ci-dessous :

%CD% - se développe en la chaîne du répertoire en cours.

%DATE% - se développe en la date actuelle en utilisant le même format que la commande DATE.

%TIME% - se développe en l'heure en cours en utilisant le même format que la commande TIME.

%RANDOM% - se développe en un nombre aléatoire compris entre 0 et 32767.

%ERRORLEVEL% - se développe en la valeur en cours de ERRORLEVEL

%CMDEXTVERSION% - se développe en le numéro de version des extensions du processeur de commande en cours.

%CMDCMDLINE% - se développe en la ligne de commande original qui a appelé le processeur de commande.

• Expansion des capacités de la commande FOR *(Commentaires extraits de l'aide en ligne sur la commande)* :

```
FOR %variable IN (ensemble) DO commande [paramètres]
```

`%variable` : Spécifie un paramètre remplaçable par une seule lettre.

`(ensemble)` : Ensemble de fichiers. Caractères génériques autorisés.

`commande` : Commande à exécuter pour chaque fichier.

`paramètres` : Liste des paramètres ou des options pour la commande spécifiée.

Pour utiliser la commande FOR dans un programme de commandes, spécifiez %%variable au lieu de %variable. Les noms de variables respectent la casse, donc %i est différent de %I.

Si les extensions de commandes sont activées, les formes supplémentaires suivantes sont prises en charge pour la commande FOR :

```
FOR /D %variable IN (ensemble) DO commande [paramètres]
```

Si ensemble contient des caractères génériques, alors la correspondance se fait sur les noms de répertoires au lieu des noms de fichiers.

```
FOR /R [[lecteur:]chemin] %variable IN (ensemble) DO commande
[paramètres]
```

Parcourt l'arborescence de répertoires depuis la racine [lecteur:]chemin, en exécutant FOR dans chaque répertoire de l'arborescence. Si aucun répertoire n'est spécifié après /R alors le répertoire en cours est utilisé. Si ensemble est seulement un point (.) alors seule l'arborescence de répertoires sera énumérée.

```
FOR /L %variable IN (début,pas,fin) DO commande [paramètres]
```

L'ensemble est une séquence de chiffres allant de début à fin, incrémenté de pas. Ainsi (1,1,5) génère la séquence 1 2 3 4 5 et (5,-1,1) génère la séquence (5 4 3 2 1)

```
FOR /F ["options"] %variable IN (ensemble-fichiers) DO
commande [paramètres]
FOR /F ["options"] %variable IN ("chaîne") DO commande
[paramètres]
FOR /F ["options"] %variable IN ('commande') DO commande
[paramètres]
```

ou, si l'options `usebackq` est utilisée :

```
FOR /F ["options"] %variable IN (ensemble-fichiers) DO
commande [paramètres]
FOR /F ["options"] %variable IN ('chaîne') DO commande
[paramètres]
FOR /F ["options"] %variable IN (`commande`) DO commande
[paramètres]
```

`ensemble-fichiers` est un ou plusieurs noms de fichiers. Chaque fichier est ouvert, lu et traité avant de passer au fichier suivant de `ensemble-fichiers`. Le traitement consiste à lire dans le fichier, le découper en lignes individuelles de texte puis analyser chaque ligne en zéro ou plusieurs parties. Le corps de la boucle FOR est ensuite appelé avec la ou les valeurs de variables prenant la valeur de la ou des parties trouvées. Par défaut, /F transmet la première partie séparée par un blanc dans chaque ligne de chaque fichier. Les lignes vides sont ignorées. Vous pouvez outrepasser le comportement d'analyse par défaut en spécifiant le paramètre optionnel "options". Ceci est une chaîne entre guillemets contenant un ou plusieurs mots-clés spécifiant diverses options d'analyse. Les mots-clés sont :

eol=c - spécifie un caractère de commentaire de fin de ligne (un seul)

skip=n - spécifie le nombre de lignes à ignorer en début de fichier.

delims=xxx - spécifie un ensemble de délimiteurs. Ceci remplace l'ensemble de délimiteurs par défaut qui sont l'espace et la tabulation.

tokens=x,y,m-n - spécifie les parties de chaque ligne devant être transmises au corps de FOR à chaque itération. Ceci provoquera l'allocation de noms de variables supplémentaires. La forme m-n est une étendue spécifiant les parties allant de m à n. Si le dernier caractère de la chaîne tokens= est une astérisque, alors une variable supplémentaire est allouée et reçoit le texte restant dans la ligne suivant la dernière partie analysée.

usebackq - spécifie que la nouvelle sémantique est en place, lorsqu'une chaîne entre guillemets inversés est exécutée en tant que commande et une chaîne entre guillemets simples est une chaîne de commande littérale et permet l'utilisation de guillemets doubles pour citer des noms de fichiers.

Quelques exemples explicatifs :

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (monfich.txt) do
@echo %i %j %k
```

…analyse chaque ligne de monfich.txt, en ignorant les lignes commençant par un point-virgule, en transmettant les 2nde et 3ème parties de chaque ligne au corps de FOR, les parties étant délimitées par des virgules et/ou espaces. Notez que le corps de FOR référence %i pour l'obtention de la 2nde partie, %j pour l'obtention de la 3ème partie et %k pour l'obtention des parties restantes après la 3ème. Pour les noms de fichiers contenant des espaces, placez les noms de fichiers entre guillemets doubles. Afin d'utiliser ainsi les guillemets doubles, vous devez également utiliser l'option `usebackq`, faute de quoi les guillemets doubles seraient interprétés comme définissant une chaîne littérale à analyser.

%i est déclarée explicitement dans la déclaration FOR et %j et %k sont déclarées implicitement via l'option `tokens=`. Vous pouvez spécifier jusqu'à 26 paries via la ligne `tokens=`, tant que cela ne provoque pas de tentative de déclaration de variable plus élevée que la lettre 'z' ou 'Z'. Souvenez-vous que les variables de FOR se composent de lettres, tiennent compte de la casse, sont globales et que plus de 52 ne peuvent pas être actives à la fois.

Vous pouvez aussi utiliser la logique d'analyse FOR /F sur une chaîne en plaçant `ensemble-fichiers` entre guillemets entre les parenthèses, en utilisant des guillemets simples. Elle sera traitée comme une ligne d'entrée simple provenant d'un fichier puis analysée.

Enfin, vous pouvez utiliser la commande FOR /F pour analyser les données en sortie d'une commande. Faites ceci en plaçant des guillemets inversés autour de `ensemble-fichiers` entre les parenthèses. Elle sera traitée comme une ligne de commande transmise à un CMD.EXE enfant et la sortie est gardée en mémoire et analysée comme s'il s'agissait d'un fichier. Ainsi l'exemple suivant :

```
FOR /F " usebackq delims==" %i IN (`ensemble`) DO @ echo
%i
```

…énumérerait les noms de variables d'environnement de l'environnement en cours.

De plus, la substitution de références de variables FOR a été améliorée. Vous pouvez maintenant utiliser la syntaxe optionnelle suivante :

```
%~I         - étend %I en supprimant les guillemets (")
%~fI        - étend %I en nom de chemin d'accès reconnu
%~dI        - étend %I en lettre de lecteur uniquement
%~pI        - étend %I en chemin d'accès uniquement
%~nI        - étend %I en nom de fichier uniquement
%~xI        - étend %I en extension de fichier uniquement
%~sI        - chemin étendu contenant uniquement des noms courts
%~aI        - étend %I en attributs du fichier
%~tI        - étend %I en date/heure du fichier
%~zI        - étend %I en taille du fichier
%~$PATH:I   - parcourt les répertoires de la variable
              d'environnement PATH et étend %I en nom du
              premier fichier reconnu trouvé. Si le nom de la
              variable d'environnement n'est pas défini ou
              que le fichier n'est pas trouvé par la recherche,
              alors ce modificateur étend en chaîne vide
```

Vous pouvez combiner les modificateurs pour obtenir des résultats composés :

```
%~dpI       - étend %I en lettre de lecteur et chemin d'accès uniquement
%~nxI       - étend %I en nom de fichier et extension uniquement
%~fsI       - étend %I en nom de chemin complet avec noms courts
              uniquement
%~dp$PATH:i - parcourt les répertoires listés dans la variable
              d'environnement PATH à la recherche de %I et étend
              en lettre de lecteur du premier trouvé.
%~ftzaI     - étend %I en DIR comme ligne en sortie
```

Dans les exemples ci-dessus %I et PATH peuvent être remplacés par d'autres valeurs valides. La syntaxe %~ se termine par un nom de variable FOR valide. Le choix de noms de variables en majuscules comme %I facilite la lecture et empêche toute confusion avec les modificateurs qui ne tiennent pas compte de la casse.

- Définition des variables d’environnement : n’ayant plus de « Autoexec.bat », les variables d’environnement sont définis via : Panneau de Configuration > Système > Avancé > Variables d’environnement.

Voilà pour ce qui est nouveautés. Il existe bien d’autres différences d’intégrations, mais seule les majeures et les plus utilisées ont été abordées.

ANNEXE 2] Mots clés ou notions à connaître abordés dans ce tutoriel

>Système d'exploitation DOS

Le DOS Il existe deux types d'exploitations de MS-DOS (**M**icro**s**oft **D**isk **O**pperating **S**ystem) est un logiciel d'exploitation (sorte de « super programme » faisant le lien entre vous et l'ordinateur) relativement ancien, orienté ligne de commande.
Cela signifie que pour communiquer avec lui il faudra connaître des mots clés ainsi qu'une syntaxe spécifique, ce qui évidemment maintenant, avec l'ère des OS graphiques (icônes, pointer-cliquer, etc…) fait de lui un système un peu obsolète.
Néanmoins, fourni avec Windows, émulé ou exécuté en mode réel, il constitue un monde intéressant pour exécuter certains programmes ou exécuter des commandes automatisées avec des scripts batch.

>Fichier

Un fichier est un ensemble de données variables enregistrées sur un périphérique de stockage. La taille et la nature du contenu des fichiers varient suivant le type de contenu, l'application créatrice ainsi que l'utilisation que l'on peut en faire.
Un fichier peut contenir par exemple une image, un texte, un son, une animation ou bien encore les quatre en même temps.
Pour des raisons techniques, le DOS « réel », c'est-à-dire celui exécuté lors du démarrage en *mode MS-DOS* d'une machine sous Windows ne gère que des noms de fichiers de 8 caractère. Par exemple, alors que sous Windows un nom de fichier « Lettre à Elise » serait correct, il faudra vous contenter de « LETTRE~1 » sous DOS. L'espace est également proscrit ainsi que d'autres caractères comme « * » ou « / » qui servent en interne à MS-DOS.
Le tilde (« ~ ») sert à une éventuelle « discrimination » des noms fichiers longs automatiquement raccourcis par le DOS : en effet, « Lettre à Elise 1 » et « Lettre a Elise 2 » se transformeraient naturellement tous deux sous DOS en « LETTREAE ». MS –DOS nommera donc « LETTRE~1 » le premier fichier, et « LETTRE~2 » le second fichier. Le nom n'excède pas 8 caractères, et les deux fichiers sont distincts.
Enfin, MS-DOS (comme Windows) identifie le type de fichier par une extension, c'est-à-dire un code à 3 caractère séparé du nom de fichier par un point. Par exemple, un fichier d'extension « txt » représente un fichier texte, un fichier d'extension « doc » un document créé avec Microsoft Word, et un fichier « bmp » une image bitmap.
Ces extensions sont réservées pour un type de données définies, c'est pour cela qu'il est déconseillé de les modifier manuellement (la modification de l'extension ne changeant évidemment pas le type du contenu du fichier !)
La longueur totale d'un nom de fichier sous MS-DOS est donc de 11 caractères (on pourra éventuellement parler de format 8+3).
Voici quelques noms de fichiers valides : « TEXTE.TXT » (fichier TeXTe), « MSDOS.SYS » (fichier SYStème), « MSPAINT.EXE » (programme EXEcutable), etc.

>Dossiers (ou répertoires)

Les répertoires sont un moyen élémentaire de regrouper, de classer et de hiérarchiser tous les fichiers. Comme dans une grande armoire que serait l'ordinateur, l'utilisateur (vous) pourrait classer ses documents (fichiers) dans autant de chemises (dossiers) qu'il le souhaiterait.

Ainsi, vous pourrez mettre tous les fichiers portant sur un même sujet dans un même dossier, de nom approprié. Bien sûr, le nom des dossiers et leur hiérarchie restent entièrement définissables par vous-même, et vous pouvez virtuellement « imbriquer » autant de dossiers que vous le souhaitez. Par exemple, le fichier « IMPOTS.DOC » représentant vos impôts payés en 1994 pourrait être classé dans le dossier « DOCS » lui-même contenu dans le dossier « IMPOTS », lui-même contenu dans le dossier « ARCHIVES » d'un disque de stockage.

>Disques et lecteurs

Toutes les données sur lesquelles vous travaillez doivent êtres stockées sur un disque de stockage, c'est-à-dire un support permanent permettant l'écriture, la lecture et la sauvegarde de votre travail.

MS-DOS ne représente pas les disques, mais leurs lecteurs. Ainsi, il fait correspondre des lettres aux lecteurs installés sur votre machine.

Par convention, le lecteur de disquettes est appelé « a », le disque dur principal « c ». D'autres lecteurs peuvent coexister, comme un second lecteur de disquette en « b » ou un lecteur de CD-ROM en « e », etc…

Avant d'accéder à vos fichiers, vous devrez donc toujours spécifier un nom de disque pour que MS-DOS puisse situer le périphérique sur lequel vous souhaitez travailler.

Le nom des lecteurs est toujours suivi de deux points. Ainsi, le lecteur de disquettes est accessible en « a : », et le disque dur en « c : », etc.

Certains médias exploités par les lecteurs peuvent être en lecture seule ou protégés en écriture, ce qui signifie que vous ne pourrez que lire les données qu'ils contiennent, et non écrire quelque chose par-dessus.

En outre, vous devez veiller dans le cas de lecteurs amovibles (a: par exemple) à ce qu'ils possèdent un disque si vous souhaitez sauvegarder des données dans ce dernier, au risque de provoquer une erreur de périphérique.

>Arborescence

L'arborescence est l'ensemble de tous les répertoires d 'un même disque.

La « base » d'un disque (c'est-à-dire un niveau où nous n'appartenons à aucun répertoire) est appelé « racine » de ce disque, et symbolisée par un « \ ». L'arborescence est donc l'ensemble des répertoires à partir de la racine d'un disque.

L'arborescence est souvent représentée de manière schématique, afin de visualiser niveaux et sous-niveaux d'un disque.

Ainsi, un disque d : contenant quatre répertoires : « DOS », « DOCS », « ARCHIVES », et « DIVERS », dont le dossier « DOCS » contenant lui-même un sous-dossier « TEST » contenant lui-même un autre dossier « FILES » sera affiché à l'écran à l'aide de la commande DOS *tree* de cette manière :

D:\
| DOS
| ARCHIVES
| DIVERS
| DOCS
| ---- TEST
| |------ FILES

>Chemin d'accès

Pour accéder à un fichier sur un disque, il ne suffit pas de connaître juste son nom, il faut aussi, dans la plupart des cas connaître sa localisation exacte dans l'arborescence.
Un fichier texte « FICHIER.TXT » sur « c : », dans le répertoire « TEST » contenu lui-même dans « DEMO » aura un chemin d'accès correspondant à la concaténation des noms de dossiers et du disque, séparés par des délimiteurs « \ » :
C:\TEST\DEMO\FICHIER.TXT

>Commandes

Pour que MS-DOS « comprenne » ce que vous voulez faire, il faudra communiquer avec lui par des mots standardisés, analysés par un programme nommé « interpréteur ».
Étant des mots clés réservés, aucun fichier ou dossier ne doit se nommer par le nom d'une commande. De plus, les commandes existent souvent sous deux formes : abrégées et entières (exemple : *MKDIR* et *MD*)

Une **commande interne** est un mot reconnu par l'interpréteur qui constitue un élément du langage MS-DOS.

Une **commande externe** est en réalité un programme fourni avec l'interpréteur. Il est placé dans un répertoire spécifique afin que MS-DOS puisse le reconnaître comme un élément du langage. Il permet une évolution, voire une meilleure souplesse du langage.
Par exemple, la commande « s*ubst* » qui permet d'affecter un nom d'unité logique (lettre de lecteur) à un répertoire d'un disque, est en fait un programme exécutable.

>DOS réel, DOS émulé

Il existe deux types d'exploitations de MS-DOS :

• Le mode *réel*, c'est-à-dire où MS-DOS est le seul maître à bord sur votre système. Il s'adresse directement à votre processeur, et gère tout seul votre mémoire. Ce mode est exécuté lors d'un *démarrage en mode MS-DOS,* et dans ce cas, Windows se retire presque entièrement de la mémoire ; ou mieux, lors du *démarrage de Windows à l'allumage de l'ordinateur*. Dans les nouvelles versions de Windows (Me, XP), ce mode n'existe plus.

• Le mode *émulé* (ou mode protégé) : C’est lorsque vous voyez les fameuses « fenêtres MS-DOS » dans Windows. En fait, ce n’est pas le vrai DOS, ce n’est qu’une simulation du DOS réel, agrémenté de nouvelles fonctions (autant de machines DOS virtuelles que vous les souhaitez, par exemple). Ce mode s’avère naturellement problématique pour certains programmes DOS spéciaux, dans ce cas, il vaut mieux recourir au mode réel)

Tutoriel Batch

http://www.astase.com

Merci à Romain TARTIÈRE
http://www.programmeurfou.fr.st