Quelques informations indispensables
Malheureusement, la majorité des sources intéressantes qui pourront vous aider à approfondir les techniques de la programmation Batch sont souvent en anglais...
"Xset" est un outil indispensable pour appendre à écrire des fichiers Batch sous toutes les versions de Windows. De plus, Marc Stern propose de nombreux didacticiels à partir de son site Internet : http://xset.tripod.com.
Jerold Schulman qui m'a autorisé à citer quelques-uns de ces scripts anime un site Internet qui propose d'innombrables pages sur la programmation Batch : http://www.jsiinc.com. C'est, par ailleurs, un des meilleurs sites traitant des systèmes NT de Windows.
D'excellents scripts sont également visibles à partir de cette adresse : www.uwasa.fi/~ts/http/http2.html#batch. Timo Salmi propose une multitude de solutions permettant de résoudre les problèmes les plus courants quand on commence à se lancer dans la programmation Batch.
Ritchie Lawrence met en lignes de nombreux modèles de scripts et propose quelques utilitaires fort bien faits qui vous permettront de réaliser des choses étonnantes en mode Ms-Dos ou en Invite de commandes. L'adresse de son site est la suivante : http://www.commandline.co.uk/index.html.
Enfin, une source inépuisable de trouvailles sont proposées à partir de ce forum recouvrant une multitude de thématiques dont Ms-Dos et l'Invite de commandes : http://www.experts-exchange.com/Operating_Systems/MSDOS.
La plupart des exemples de scripts expliqués dans ce chapitre s'inspirent directement ou indirectement des techniques présentées par ces différents auteurs.

Créer un fichier Batch
Le terme "Batch" désigne un fichier contenant une suite de commandes qui seront traitées automatiquement. Nous appelons aussi cela un "traitement par lot". Afin de créer votre premier fichier Batch suivez cette procédure :
1) Ouvrez un éditeur de texte : le Bloc-notes par exemple.
2) Inscrivez vos lignes de commandes.
3) Enregistrez votre fichier texte.
4) Cliquez sur le nom du fichier puis appuyez sur la touche F2.
Vous serez en mode "Édition".
5) Renommez le fichier en changeant l'extension .txt en .bat.
À la question : "Voulez-vous vraiment renommer le fichier", répondez par Oui.
Attention de désactiver au préalable la case "Masquer les extensions de fichiers dont le type est connu" dans les options avancés de l'Explorateur Windows.

Quelle différence entre l'extension .cmd .btm et .bat ?
Un fichier .cmd ou .btm ne sera pas reconnu en tant que tel par Windows 9X. Nous pouvons voir cela comme une sorte de garde-fou si tel fichier de

commande est parfaitement incompatible avec ces versions de Windows. Par ailleurs, le processus est légèrement différent :
Dans le cas d'un fichier .bat ou .cmd, chaque ligne du fichier est exécutée individuellement et le fichier fermé puis ouvert à chaque lecture d'une nouvelle commande. Dans le cas d'un fichier .btm, le fichier n'est ouvert qu'une fois, puis lu en mémoire et enfin fermé. C'est donc a priori le mode le plus rapide. surtout si ce sont des commandes internes qui sont exécutées.

Savoir se servir de la commande Echo
Dans un nouveau fichier Batch, copiez ce contenu :

```
echo La commande Echo est active
@echo
date /t
echo off
date /t
@echo off
date /t
```

L'utilisation de l'arobase permet de biffer le statut de la commande "Echo". Par ailleurs, la commande "Echo off" évite l'affichage des commandes contenues dans le script.

Se servir des parenthèses dans les fichiers de scripts
Si vous souhaitez rediriger le résultat de différentes commandes dans un même fichier texte. Par exemple :

```
dir /s /b *.doc >> résultat.txt
dir /b *.dot >> résultat.txt
etc.
```

Il est dans ce cas plus simple de saisir :

```
(
dir /s /b *.doc
dir /s /b *.dot
) >> résultat.txt
```

Cela vous évitera de spécifier plusieurs fois le même fichier de sortie. Par ailleurs, c'est une manière de créer de toutes pièces un fichier texte :

```
@echo off
(
@echo Bonjour,
@echo Tout le monde !
```

) > test.txt

Dans ce dernier cas, les commandes sont regroupées.

Faire des remarques
La commande "Rem" vous permet d'insérer des commentaires ou de désactiver temporairement des commandes inclues dans votre fichier batch. Ce n'est pas tout à fait vrai. La commande rem echo quelque chose > Sortie.txt ne fait qu'envoyer rien au fichier Sortie.txt. La commande est donc exécutée mais en mode "désactivé". Une sorte de coup à blanc... Pour les amoureux de la performance il est donc plus judicieux d'utiliser :: plutôt que Rem. Les deux points étant considéré comme l'indication d'une étiquette la commande ne sera pas exécutée. À titre de test saisissez tour à tour ces deux commandes :
rem echo A
:: echo B
La seconde commande provoque un retour chariot et non un saut de ligne. Dans un fichier Config.sys il est possible de désactiver une commande en utilisant un point virgule : ;connexion=c:\dos\ramdrive.sys

Afficher le code de sortie
À chaque fois que vous saisissez une commande, cette dernière produit un code de sortie qui sont principalement les suivants :
1 : la commande a renvoyé une erreur
0 : la commande n'a pas renvoyé d'erreur.
C'est la variable %Errorlevel% qui est chargée de suivre les fluctuations de la réussite ou non des commandes exécutées.
Information : La commande "Set" teste les "Errorlevel" en partant de la plus petite valeur puis en procédant par incrémentation de 1. La commande "Goto" démarre de la plus grande valeur puis procède par décrémentation de 1.

Se servir des étiquettes
La commande "Goto" vous permet d'atteindre un point précis de votre script. La commande spécifiée doit porter un titre que nous appelons une étiquette. Cette ligne de titre commence obligatoirement par deux points. Nous sommes obligés d'anticiper quelque peu sur l'explication des conditions car c'est une des utilisations privilégiées des étiquettes. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
dir *.doc
if not errorlevel 1 goto Fin else if goto Avertissement
:Avertissement
```

```
echo Aucun fichier .doc !
:Fin
```

La variable %Errolevel% nous permet de saisir si la commande "Dir" a renvoyé une valeur 1 ("Échec") ou 0 ("Opération réussie"). Si au moins un fichier .doc est trouvé nous allons directement à l'étiquette :Fin. Ou, plus exactement si la sortie d'erreur n'est pas 1 ("Échec"), alors il faut se rendre à l'étiquette :Fin. Sinon (else if) nous nous rendrons à l'étiquette :Avertissement. Si nous rajoutons une commande à la suite de l'étiquette :Fin, cette dernière sera automatiquement exécutée même si la condition n'est pas remplie. Afin d'éviter cela, il nous faut nous servir d'une étiquette spéciale : goto:eof. Notre fichier script devient alors :

```
@echo off
dir *.doc
if not errorlevel 1 goto Fin else if goto Avertissement
:Avertissement
echo Aucun fichier .doc ! & goto:eof
:Fin
echo Processus fini !
```

Dans ce dernier cas, la commande goto:eof nous permet d'effectuer une césure dans le script. Cela revient à dire : "Affiche le message 'Aucun fichier .doc !' puis ne fait plus rien". En termes savants, puisque nous ne définissons pas d'étiquette, nous transférons le contrôle à la fin du script en cours.

Gérer les caractères accentués

Les différences entre la norme Ansi (Windows) et OEM (Ms-Dos) font que si vous vous servez d'un éditeur de texte comme le Bloc-Notes les caractères accentués ne s'afficheront pas correctement dans les sorties écran en mode "Console". Ainsi, la phrase "Je répète : "Vous êtes un entêté"" s'affichera comme cela : Je rÚpÞte : "Vous Ûtes un entÛtÚ". Il y a différentes solutions : Vous pouvez toujours vous servir de l'éditeur de texte prévu dans toutes les versions de Windows : "Edit".

ll existe de nombreux programmes vous permettant d'opérer la conversion Ansi vers OEM et vice-versa. "OEMANSI.exe" est un programme écrit en français que vous pouvez télécharger à partir de cette adresse : http://perso.wanadoo.fr/andre.araste/tele2.htm. Décompressez l'archive ZIP nommée Oemansi.zip puis double-cliquez sur ce fichier exécutable : Setup.exe afin de lancer l'installation de ce logiciel.

Imaginons que notre fichier à convertir s'appelle Test.bat suivez la procédure suivante :

1) Cliquez sur Démarrer/Tous les programmes/OEMANSI/OEMANSI.
2) Cliquez sur le bouton Ouvrir Fichier.

3) Dans la liste déroulante Fichiers de type :, sélectionnez Tous les fichiers (*.*) puis votre fichier Batch.
4) Cochez le bouton radio ANSI(Win) puis la flèche dirigée vers le bas.
Il est possible d'effectuer l'opération inverse !
5) Cliquez sur le bouton Sauvegarde OEM puis lancer le processus de conversion.
6) Confirmez le remplacement ou non de l'ancienne version.

Gérer les caractères réservés
Ces caractères ne peuvent pas être employés "tel quel" dans un script. Le signe % pour être "compris" dans un fichier de script doit être redoublé. Par ailleurs, nous avons déjà vu que pour afficher un caractère réservé il faut le faire précéder du signe ^. Dans ce cas là, il ne sera pas assimilé à une commande mais bien à un caractère. Afin d'afficher les signes : ! ^ & < > >> " | créez un fichier Batch contenant cette commande :
```
echo ^! ^% ^^ ^& ^< ^> ^>^> ^" ^|
```
Une manière d'afficher les caractères de redirection :
```
@echo off
<nul (set /p z=Le^|texte^|de^|sortie) >sortie.txt
```
Nous redirigeons le produit de la commande "Set" vers le fichier Sortie.txt mais en désactivant toute sortie écran. L'utilisation de la commande "Set" est expliquée juste après.

Manipuler les variables en Invite de commandes
L'Invite de commandes a prévu un certain nombre de variables internes :

Nom de la variable - Fonction
%CD% Affiche le répertoire courant.
%DATE% Affiche la date.
%TIME% Affiche l'heure.
%RANDOM% Affiche une valeur décimale comprise entre 0 et 32767.
%ERRORLEVEL% Affiche le code d'erreur renvoyée par la précédente commande.
%CMDEXTVERSION% Affiche la version du processeur de commande.
%CMDCMDLINE% Affiche la ligne de commande originelle invoquée par le processeur de commande.

La commande Set p affiche toutes les variables commençant par la lettre P.
Une suite possible de commandes est :
```
set variable=c:\test
set variable
```
Nous pouvons vérifier que notre variable est bien égale à C:\Répertoire. Nous pouvons également saisir :
```
dir %variable%
```

Le contenu de l'arborescence C:\Test s'affichera.
Une autre manière consiste à saisir le nom de l'arborescence à la suite du nom du fichier de commande. Par exemple : batch c:
Le script nommé Batch permettant de lister les fichiers texte du répertoire indiqué aura alors ce contenu :
dir %1%*.txt
Nous pouvons procéder à l'extraction de tous les caractères à l'exception des trois premiers. Saisissez donc :
set 1=dir *.txt
echo %1:~3%
echo %1:~-3%
Seuls les trois derniers caractères seront affichés.
echo %1:~0,-4%
Tous les caractères seront affichés à l'exception des 4 derniers.
echo %1:~0,3%
Nous afficherons les trois premiers caractères de la variable passée précédemment.
echo %1:~4,5%
Dans ce dernier cas, 5 caractères seront affichés en partant du 4ème caractère.
Attention : si vous utilisez un caractère réservé dans votre variable, ce dernier doit être "signalé". Afin de passer une variable nommée Test&1 il vous faudra saisir : set x =test^&1
Si vous ne faites pas précéder le caractère réservé du signe ^ vous aurez droit à un message d'erreur salé !
Par exemple, saisissez ces commandes :
set x=%time%
set y=%date%
echo %x% %y%
echo %x:~0,5% %y%
Nous reprenons deux variables internes pour définir deux nouvelles variables nommées x et y.
Il est possible de cacher tous les répertoires en saisissant cette commande :
set dircmd=0
Le contenu des arborescences restera invisible bien que leur accès reste possible. Afin de revenir à la situation précédente, saisissez :
set dircmd=
Astuce : Il est possible d'assigner à une variable un nombre aléatoire.
Saisissez ces commandes :
set variable=%random%
echo %variable%
Tilmo Salmi sur cette page web : www.uwasa.fi/~ts/http/http2.html#batch signale un problème concernant l'utilisation de la commande "Echo" quand cela s'applique aux variables. Comparez la sortie écran de ces deux fichiers

Batch :

```
@echo off
set variable=Ceci est un test
echo %variable%
set variable=
echo %variable%
@echo off
set variable=Ceci est un test
echo. %variable%
set variable=
echo. %variable%
```

Seul le second fichier Batch n'affiche pas d'erreur. L'explication est simple : la commande "Echo" suivie d'un point force l'affichage d'une ligne vide.

"Transtyper" une variable

Il est donc possible de modifier une variable en additionnant à sa définition toute sorte de chaîne de caractères. Cette opération s'appelle du "transtypage". Par exemple, saisissez ces commandes :

```
set datation =La date d'aujourd'hui est le %date%
set datation
```

Créez un nouveau fichier Batch contenant :

```
@echo off
set chaîne=[Ceci n'est qu'un test]
set chaîne=%chaîne:[=%
set chaîne=%chaîne:]=%
echo %chaîne%
```

Dans cet exemple, nous ne faisons que dire qu'aux caractères [et] rien ne correspond (et donc ils ne s'afficheront pas).
Le script suivant affichera : (Ceci n'est qu'un test)

```
set chaîne=[Ceci n'est qu'un test]
set chaîne=%chaîne:[=(%
set chaîne=%chaîne:]=)%
echo %chaîne%
```

Nous pouvons remplacer la deuxième et troisième ligne par :

```
set chaîne=%chaîne:~1,21%
```

Cela appelle deux remarques :
* La numérotation commence à partir de zéro : au caractère [correspond donc le numéro 0.

* Les espaces sont comptés.
Afin de convertir une chaîne de caractères de minuscules en majuscules :

```
@echo off
set chaîne=%1
set chaîne=%chaîne:a=A%
set chaîne=%chaîne:b=B%
set chaîne=%chaîne:c=C%
::Vous pouvez indiquer toutes les lettres de l'alphabets...
set chaîne=%chaîne:y=Y%
set chaîne=%chaîne:z=Z%
echo %chaîne%
```

Nous passons dans un premier temps une variable nommé %chaîne% qui sera la chaîne de caractère saisie par l'utilisateur ou, si nous préférons, le premier paramètre saisi à la suite du nom du fichier Batch : %1.
Le reste du script opère la substitution des variables en changeant à chaque fois la casse de la lettre.

Créez un formulaire permettant de récupérer une ou plusieurs variables
En Invite de commandes la solution est assez simple... Saisissez ces commandes :

```
set /p Variable=Veuillez entrer votre nom:
set variable
```

Il est donc possible de se servir de "Set" en lieu et place de "Choice". Créez un nouveau fichier Batch en copiant les lignes suivantes :

```
@echo off
:début
echo 1 : Menu1
echo 2 : Menu2
echo 3 : Menu3
set /p choix=choisissez un chiffre.
if not %choix%==" set choix=%choix:~0,1%
if %choix%==1 goto Menu1
if %choix%==2 goto Menu2
if %choix%==3 goto Menu3
echo %choix% n'est pas bon !
goto début
:Menu1
echo Bonjour & goto:eof
:Menu2
echo Au revoir & goto:eof
```

```
:Menu3
echo Adieu & goto:eof
```

Les quatre premières lignes affiche un formulaire. La quatrième ligne nous permet d'attendre que l'utilisateur ait saisie la chaîne qui sera associée à la variable %choix% que nous allons définir. Dans ce cas le commutateur /p est obligatoire.
Si l'utilisateur appuie sur la touche Entrée sans avoir saisi de chiffre, la variable %choix% sera défini sur le chiffre 2. Pour vous en rendre compte saisissez ces commandes :

```
set choix=%choix:~0,1%
echo %choix%
```

Dans ce cas, nous nous rendons directement à l'étiquette n°2 : Menu2.
Si le chiffre saisi n'est ni 1, 2 ou 3 alors il sera affiché un message d'erreur puis nous retournerons en début de script à l'étiquette :début.
Chaque menu affiche un message puis termine l'exécution du script par la commande goto:eof.
Nous savons donc qu'il est possible de spécifier plusieurs conditions "à la chaîne". Il suffit de dire que si aucune des trois premières conditions n'a été remplie alors c'est que la saisie de l'utilisateur n'est pas valable. C'est la seule petite astuce de ce script !

Faire des calculs en Invite de commandes
Une possibilité consiste à utiliser la commande "Set". Saisissez ceci :

```
set /a z=356/356/1*356
```

Le tableau suivant présente les opérateurs autorisés avec le commutateur /a par ordre de priorité décroissante :

Opérateur - Opération effectuée
< > Groupement
* / % + - Calcul
<< >> Décalage logique
& ET au niveau du bit
^ OU exclusif au niveau du bit
| OU au niveau du bit
= *= /= %= += -= &= ^= |= <<= >>= Attribution
, Séparateur d'expression

Toute chaîne non numérique est traitée comme un nom de variable dont les valeurs sont converties en nombre avant d'être utilisées.
Si une variable est traitée sans être définie dans l'environnement en cours, sa valeur sera égale à zéro. L'ajout du signe % est donc inutile.
Il est facile d'imaginer une variable qui soit celle-ci : 0010. Nous pouvons supprimer les zéros placés devant en saisissant cette commande :

set /a variable=100%variable%%100
Avec l'opérateur % est effectuée une division modulo. Dans une division modulo, les deux valeurs sont divisées mais le résultat n'est que le reste de la division. Si vous saisissez : 19 % 5, vous obtenez comme résultat modulo 4, parce que 19 divisé par 5 égal 3 reste 4. Afin de le vérifier, saisissez ces commandes :
set /a variable=19
set /a variable=%variable%%5
Par ailleurs, si vous saisissez cette commande :
set /a variable=0010
Vous obtenez comme résultat le chiffre 8. En fait, les nombres sont en notation décimale à moins qu'ils ne soient préfixés par 0x pour les valeurs hexadécimales ou par 0 pour la notation octale. Il est donc possible d'obtenir instantanément la valeur décimale d'un chiffre hexadécimal. À titre de test, saisissez ceci : set /a 0x3e7.
Astuce : Ce type de commande marche : (set /a 75 /15) > sortie.txt.
Voici un exemple de script tout simple :

```
@echo off
call :fonction %1 %2
echo La somme est %somme%
call :soustraction %1 %2
echo La soustraction est %soustraction%
goto :eof
: fonction
set /a somme=%1+%2
: soustraction
set /a soustraction=%1-%2
```

Saisissez le nom de votre fichier Batch suivi des deux éléments de l'opération. Nous récupérons les deux paramètres en nous servant des variables %1 et %2. La commande "Set /a" nous permet de calculer la valeur respective des variables %somme% et %soustraction%.
Justifier un nombre inclus dans une variable
Dans un nouveau fichier batch ciopiez ce contenu :

```
@echo off
set nombre=%1
if %1 lss 10000 set a=
if %1 lss 1000 set a=0
if %1 lss 100 set a=00
if %1 lss 10 set a=000
set nombre=%a%%nombre%%
set nombre=%nombre:"=%
```

```
echo %nombre%
```

Saisissez le nom de votre fichier batch suivi du nombre à justifier. Nous récupérons le paramètre saisi par l'utilisateur en vérifiant si le nombre et inférieur à 10000 puis à 1000 et ainsi de suite. À chaque fois nous définissons une variable nommé %a% que nous ferons précédée à la variable %nombre%. L'avant dernière ligne nous permet simplement de supprimer les guillemets de la variable obtenue.

Localiser les variables d'environnement

Les modifications d'environnement définies par la commande "Setlocal" ne sont appliquées localement qu'à votre fichier de commandes. Lorsqu'une commande "Endlocal" est rencontrée, les paramètres précédents sont rétablis. Il n'est pas possible d'utiliser ces variables indépendamment d'un script ou d'un fichier de commandes.
Il nous est ainsi possible d'appeler un fichier de commandes situé à un emplacement différent de celui à partir duquel nous lançons le premier fichier de commandes. Par exemple, créez un fichier nommé A.bat contenant ces lignes :

```
@echo off
setlocal
path=F:\;%path%
call B >c:\sortie.txt
endlocal
start notepad c:\sortie.txt
```

Créez un autre fichier nommé B.bat que vous placerez sur un autre lecteur (dans cet exemple F:) et qui contiendra la commande à exécuter :

```
dir *.txt
```

Lancez le fichier A.bat...
La commande path=F:\;%path% nous permet de préciser que la variable définie %F:% vient s'ajouter aux autres chemins prédéfinis par la variable d'environnement %path%.
La commande call B appelle à partir du fichier A.bat le fichier B.bat. cela revient à lancer l'exécution d'un second script à partir d'un premier script.
Si nous avions pas ajouter le lecteur F: dans le "Path", le fichier B.bat n'aurai pas été trouvé.
Les commandes "Setlocal" et "Endlocal" ne sont là que pour éviter un éventuel conflit entre la définition d'un même nom de variable dans un même script ou entre deux scripts exécutés au cours de la même session d'Invite de commandes.

Définir différentes variables dans un fichier Batch
Imaginons que nous souhaitons pouvoir exécuter trois commandes différentes en fonction des circonstances. Dans un nouveau fichier Batch, copiez le texte suivant :

```
@echo off
if {%1}=={c} set lecteur=c:\&goto Dir
if {%1}=={d} set lecteur=d:\&goto Dir
if {%1}=={e} set lecteur=e:\&goto Dir
@echo Saisissez le nom du fichier Batch suivi d'un espace puis appuyez sur la touche C D ou E
:Dir
dir %lecteur% > %lecteur%\Sortie.txt
```

Nous nous servons du premier paramètre saisi par l'utilisateur à la suite du nom du script : %1. Si, par l'exemple, la touche appuyée est la lettre C alors nous définissons une variable nommée %Lecteur% qui sera égale à c:\ puis nous nous rendons à l'étiquette nommée :Dir.
Si la touche choisie n'est ni C, D ou E alors un message sera affiché.
Dans le cas contraire, un fichier nommé Sortie.txt sera créé à la racine du lecteur analysé.

- Autoriser ou non les variables temporisées :
Créez un premier fichier Batch contenant ceci :

```
set variable=avant
if "%variable%" == "avant" (
set variable=maintenant
if "%variable%" == "maintenant" @echo Si vous voyez ceci le script marche
)
```

Si vous l'exécutez le message ne sera pas affiché.
Nous créons une variable nommée %avant%. Si notre variable correspond à la chaîne de caractères %avant% alors nous assignons à cette variable une nouvelle chaîne nommée %maintenant%. La dernière condition pose ceci : si notre variable est égale à %maintenant% alors affiche un message. Comme la variable a été fixée sur %avant% la condition n'est pas remplie et donc le message pas affiché. Créez un autre fichier Batch avec cette fois-ci ce contenu :

```
setlocal EnableDelayedExpansion
set Variable=avant
if "%Variable%" == "avant" (
set Variable=après
```

```
if "!Variable!" == "après" @echo Si vous voyez ceci le script marche
)
```

Dans ce dernier cas les variables sont assignées dynamiquement et leur affectation vient écraser celle qui leur avait été au préalable assignée. Il y a plusieurs manières d'assigner des variables de manière dynamique :
* Insérez une variable d'environnement en utilisant la commande "Setlocal EnableDelayedExpansion".
* Lancez l'interpréteur de commandes en utilisant le commutateur /V:ON.
* Modifiez directement le Registre :
1) Cliquez sur Démarrer/Exécuter puis saisissez : regedit
2) Ouvrez HKEY_CURRENT_USER\Software\Microsoft\Command Processor.
3) Dans le volet de droite créez une nouvelle valeur DWORD nommée DelayedExpansion.
4) Éditez cette valeur et affectez comme données de la valeur le chiffre 1.
5) Quittez puis relancez "cmd".
Il est possible d'opérer la même modification dans l'arborescence HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor si vous souhaitez que les modifications soient opérationnelles pour l'ensemble des utilisateurs de votre machine. Comparez ces deux fichiers Batch :

```
@echo off
setlocal EnableDelayedExpansion
set Liste=
for %%a in (*) do set Liste=%%a& echo !Liste!
```

```
@echo off
set Liste=
for %%a in (*) do set Liste=%%a& echo %Liste%
```

Dans ce dernier cas, rien ne sera affiché.
Le premier script affichera le contenu du répertoire courant (*). Nous assignons à chaque nouvel objet trouvé (fichier ou répertoire) la variable %liste% que nous affichons en utilisant la commande "Echo".

Déplacer la position des paramètres de commandes
Dans un nouveau fichier de commandes appelé Test.bat, copiez le texte suivant :

```
@echo off
:Copie
shift
if "%1"=="" goto Fin
copy %1 c:\sauvegarde
goto Copie
```

```
:Fin
```

Saisissez par exemple : test *.txt *.doc *.bat *.bmp *.ppt *.eml
Nous allons donc copier tous les fichiers spécifiés du répertoire courant vers le répertoire C:\Sauvegarde. Le principe consiste à utiliser la commande "Shift" qui remplacera la variable %0 par la variable %1 (*.doc), la variable %1 par %2 (*.bat) une fois la première commande terminée (et ainsi de suite). En l'absence de la commande "Shift" nous chercherions désespérément un fichier introuvable... Il est donc possible de créer un fichier de commandes qui accepte plus de dix paramètres (de 0 à 9) puis que les arguments situés à partir de la dixième position seront décalés un à un afin d'occuper la variable %9.
La commande spécifiant une condition nous permet d'éviter de rentrer dans une boucle sans fin. Si la variable actuelle est la même chose que rien ("") alors nous allons à l'étiquette :Fin. Dans un premier temps, nous avons donc cette commande : if "*.txt"=="" goto fin. Comme *.txt est différent de rien, nous retournons à l'étiquette :Début. S'il n'y a plus aucun paramètre trouvé, nous poserons alors la condition suivante : if ""=="" goto Fin.
Et là, dans ce cas, le script prend fin !
Il y a une manière légèrement différente qui nous permet de spécifier le répertoire de destination. Dans un nouveau fichier Batch copiez ces lignes :

```
@echo off
set variable=%1
:Copie
shift
if "%1"=="" goto Fin
copy %1 %variable%
goto Copie
:Fin
```

Saisissez le nom de votre fichier Batch suivi du répertoire de destination et des fichiers à copier. Par exemple : test c:\sauvegarde *.txt *.xls *.doc *.bat *.bmp *.ppt *.eml
La première variable %1 passe le premier argument saisi : le répertoire de destination.
Nous définissons cette première variable sous le nom de %variable%.
L'étiquette :Copie annonce les commandes permettant le processus de copie.
La commande "Shift" décale la variable : %1 devient %0. Nous avons donc simplement défini le répertoire de destination. Pour le reste le principe de fonctionnement est strictement identique au script précédent.

Appeler un programme à partir d'un autre programme
La commande "Call" vous permet d'appeler un programme sans que le

programme "parent" soit stoppé. Créez un fichier de commandes nommé Test.bat contenant simplement ces lignes :
dir %1 /s> %2
Pour envoyer le contenu d'un répertoire dans un fichier nommé Sortie.txt, saisissez :
test . sortie.txt
La première variable %1 est donc défini sur le répertoire par défaut (.) et la seconde (%2) sur le nom du fichier de sortie (Sortie.txt). La commande fonctionne quelque soit le répertoire par défaut et le nom du fichier de sortie.
Nous aurions également pu saisir : test c:\Documents and settings\ sortie.txt
Créez un nouveau fichier de commande contenant ces lignes :

@echo Le nom du script est : %0
@echo Le nom de la variable 1 est : %1
@echo Le nom de la variable 2 est : %2
@echo Le nom de la variable: 3 est : %3

Enregistrez le sous le même nom. Saisissez maintenant :
test Jean Marc Michel
Créez maintenant un nouveau fichier de commande contenant ces lignes :

@echo off
Call :Chemin & goto:eof
:Chemin
@echo Voici le chemin: %~dp0

Ce script donne tout bêtement le chemin d'accès du répertoire courant.
La commande "Call" appelle une étiquette nommée :Chemin.
La variable %~dp0 permet d'obtenir la lettre de lecteur et le chemin de l'arborescence visitée.
L'appel vers l'étiquette :eof permet d'arrêter l'exécution du script une fois le chemin affiché.
Vous pouvez placer ce script là où vous voulez puisqu'il aura toujours la capacité de retrouver le répertoire courant et les commandes qui seront placées dans la même arborescence.
Il vous est possible de passer différentes variables si la commande principale en contient plusieurs. Créez un fichier de commande (appelé A) contenant ces lignes :
call b %1 %2
Et un second fichier (appelé B) contenant ceci :
dir %1%*.*
dir %2%*.*
Saisissez cette commande : a c: c:\temp
Le fichier de commande A appelle le fichier B en lui passant les deux

variables que vous avez saisies : %C:% et %C:\Temp%.
Par ailleurs, il est possible de se servir des commande "Call" et "Goto" pour créer des sous-routines. Vérifiez la sortie écran produite par ce "Batch" :

```
@echo off
echo commande1
call :echo2
echo commande4
goto :eof
: echo2
echo commande2
call :echo3
goto :eof
: echo3
echo commande3
goto :eof
```

L'avantage de la commande "Call" sur un simple appel d'étiquette induit par la commande "Goto" est que le déroulement du script n'est pas interrompu. Nous appelons simplement les instructions contenues sous l'étiquette puis reprenons le déroulement de notre "Batch" là où nous l'avions laissé.

Les structures de contrôles
Cette partie concerne toutes les opérations vous permettant de récupérer des paramètres, la sortie d'une commande ou les saisies effectuées par les utilisateurs.

Exécuter une commande pour un jeu de fichiers
La syntaxe générale est celle-ci :
For {%variable|%%variable} in (jeu) do commande
[Options_Ligne_Commande]
Prenons tout d'abord un exemple... Si nous souhaitons afficher le contenu de tous les fichiers .txt et .doc du répertoire courant, nous saisirons :
for %1 in (*.doc *.txt) do type %1
Signalons que la variable peut être un chiffre ou une lettre (par exemple, %a fait aussi bien l'affaire).
Astuce : Si nous utilisons cette commande dans un fichier de commande le signe % doit être redoublé.
Jeu représente les noms de fichiers et doit être placé entre parenthèses.
Les mots clés in et do sont possibles lors de l'utilisation de la commande "For". Nous pouvons les traduire par ceci : Pour tous les fichiers présents dans (in) ce jeu de fichiers exécute (do) cette commande. Il vous est possible d'utiliser les paramètres spécifiques à chaque commande (Options_Ligne_Commande).

Une manière de lister les noms seuls des fichiers exécutables du répertoire courant est de saisir :
for %a in (*.exe) do @echo %a
La simple liste de tous les fichiers (à l'exception des répertoires) s'obtient en saisissant : for %a in (*) do @echo %a

Exécuter une commande de manière récursive
En Invite de commandes si vous souhaitez afficher tous les sous-répertoires appartenant à votre profil vous saisirez :
for /r "%userprofile%" %a in (.) do @echo %a
Quand Jeu est simplement un point, c'est l'arborescence principale qui est énumérée.
Si vous souhaitez simplement lister les sous-répertoires de l'emplacement par défaut saisissez :
for /r %a in (.) do @echo %a
Afin de supprimer tous les fichiers .doc de l'emplacement par défaut il suffit de saisir :
for /r %a in (*.doc) do del /q "%a"

Gestion des noms comprenant des espaces
La commande suivante ne fonctionnera pas si le nom d'un des fichiers .doc ou .txt contient des espaces :
for %a in (*.doc *.txt) do copy %a c:\temp\
Vous devez dans ce cas mettre la seconde variable entre guillemets :
for %a in (*.doc *.txt) do copy "%a" c:\temp\

Effectuer une substitution de variable
Il est possible de changer les variables de sortie. Par exemple, si nous saisissons cette commande :
for /r %a in (*.txt) do @echo %~na
Seuls les fichiers .txt seront affichés.
La liste ci-dessous récapitule les variables de substitution ainsi que les combinaisons permises. Rappelons que la variable %a peut être remplacée par n'importe quelle autre lettre de l'alphabet ou chiffre.

%~a : développe %a en supprimant les guillemets.
%~fa : développe %a en un nom de chemin complet.
%~da : développe %a en une lettre de lecteur seulement.
%~pa : développe %a en un chemin seulement.
%~na : développe %a en un nom de fichier seulement.
%~xa : développe %a en une extension de fichier seulement.
%~sa : développe le chemin afin qu'il ne contienne que des noms courts.
%~aa : développe %a jusqu'aux attributs du fichier.
%~ta : développe %a jusqu'à la date et l'heure du fichier.

%~za : développe %a jusqu'à la taille du fichier.
%~$PATH:a : recherche les répertoires énumérés dans la variable d'environnement et développe %a jusqu'au nom complet du premier répertoire trouvé.
%~dpa : développe %a en une lettre de lecteur et un chemin seulement.
%~nxa : développe %a en nom de fichier et une extension seulement.
%~fsa : développe %a en un nom de chemin complet avec des noms courts seulement.
%~dp$PATH:a : recherche les répertoires énumérés dans la variable d'environnement pour %a et développe jusqu'à la lettre du lecteur et au chemin du premier répertoire trouvé.
%~ftzaa : développe %a en une ligne de sortie semblable à celle affichée par la commande "Dir".

Comparez les sorties affichées par ces trois commandes :
for /r %a in (*.txt) do @echo %~dpa
for /r %a in (*.txt) do @echo %~nxa
for /r %a in (*.txt) do @echo %~ftzaa
Signalons que la commande for /r %a in (*.txt) do @echo %~dp$PATH:a n'affichera que les arborescences contenant au moins un fichier Texte.
Là aussi, comparez le texte de sortie avec cette commande :
(for /r %a in (*.txt) do @echo %~$PATH:a) > Sortie.txt
Nous avons simplement mis entre parenthèse la commande et rediriger son résultat dans un fichier .txt nommé Sortie.txt.

Décomposer une variable
Dans un nouveau document Batch copiez ce contenu :
for /f "tokens=6-8 delims=/ " %%a in ('echo. ^|date') do set jour=%%a&set mois=%%b& set année=%%c
La commande "Date" affiche ce texte : "La date du jour est : 12/07/2004"
Nous devons donc récupérer le sixième bloc de caractères (12/07/2004) et, si nous enlevons les barres inversées, 8 caractères à partir de la fin du sixième bloc (12 07 2004).
Trois valeurs nous intéressent : 12, 07 et 2004 que nous transformons en variables en utilisant la commande "Set".
Pour vous en assurer saisissez cette commande : set jour
Nous pouvons faire la même opération pour la commande "Time" :
for /f "tokens=2-8 delims=:," %%a in ('echo. ^|time ^| find "actuelle"') do set HH=%%a&set HM=%%b& set HS=%%c& set HDS=%%d
Dans ce cas le principe est un peu différent :
Si je saisis la commande Time, le message suivant apparaît : "L'heure actuelle est : 11:51:29,13" - "Entrez la nouvelle heure". Évidemment seule la première ligne nous intéresse... Nous allons demander que soit afficher la ligne qui contient le mot "actuelle" : echo. ^|time ^| find "actuelle"

Dans cette ligne nous allons extraire le second bloc puis les huit caractères en partant de la fin : tokens=2-8
Du résultat brut (11:51:29,13) nous allons effacer les signes : et ,(delims=:,).
Quatre variables vont être définies : HH, HM, HS et HDS correspondant respectivement à l'heure, aux minutes, aux secondes et aux dixième de secondes. Nous devons donc spécifier quatre noms de variables : %a, %b, %c et %d.
L'opérateur & permet d'exécuter plusieurs fois la commande "Set" en une seule ligne.
Le tableau suivant récapitule les mots clés que vous pouvez utiliser :

Mot clé - Signification
eol=c : Indique un (et un seul) caractère de fin de ligne
skip=n : Spécifie le nombre de lignes à sauter à partir du début du fichier.
delims=xxx : Spécifie un séparateur.
tokens=x,y,m-n : Spécifie les jetons à passer à la commande For.
usebackq : Permet l'utilisation des guillemets.

Nous voyons plus en détails le principe d'utilisation d'un jeton au paragraphe suivant.

Principes d'utilisation d'un jeton
Le mot clé "Tokens" sert à allouer un jeton pour chaque fichier analysé. Un jeton est sorte de permission temporaire accordée à un fichier afin que soit exécuté la commande spécifiée. Une fois cette dernière achevée, le jeton est alloué au prochain fichier trouvé et ainsi de suite. Si nous souhaitons activer l'attribut "Lecture seule" des fichier .doc nous pouvons saisir :
for /f "tokens=*" %a in ('dir /s /b *.doc') do attrib +r %a
Bien entendu, notre commande s'exécutera pour tous les fichiers .doc placés dans les répertoires et sous-répertoires de l'emplacement par défaut. Nous pouvons traduire cette commande par : "Pour chaque fichier trouvé quand je lance la commande 'Dir' exécute celle qui suit le mot clé Do". Il vous est également possible d'utiliser des itérations de plages de valeurs afin, par exemple, de créer rapidement dix répertoires nommés Test et numérotés de 1 à 10 :
for /l %a in (1,1,10) do @md test%a
Nous avons rajoutez une arobase devant la commande "Md" afin de désactiver l'affichage de la sortie écran.
De la même façon vous pouvez créer des "listes négatives" :
for /l %a in (-10,1,10) do @echo %a >> sortie.xls
Un document Excel sera créé avec une plage de valeurs comprises entre -10 et 10.
La syntaxe est la suivante : Valeur_Début, Valeur_Incrémentation, Valeur_Fin.

Créez un fichier texte nommé Fichier.txt contenant ceci :

Cellule1 Cellule2 Cellule3 etc.
-Cellule4 Cellule5 Cellule6 etc.
Cellule7,Cellule8,Cellule9 etc.
Cellule10 Cellule11 Cellule12 etc.

Saisissez cette commande si vous voulez afficher l'ensemble de Fichier.txt à l'exception de la seconde ligne :
for /f "eol=- tokens=1,2* delims=, " %a in (fichier.txt) do @echo %a %b %c
"eol=-" : spécifie que les lignes commençant par un - seront ignorées.
"tokens=" : alloue un jeton puis un second pour chaque ligne explorée. Deux blocs de données seront adressées. L'astérisque permet d'allouer autant de variables que nécessaire pour recevoir le texte "restant". C'est plus simple d'utilisation que d'indiquer : "tokens=1,2,3,4" (puis que nous devons afficher quatre blocs de données).
"delims=, " : nous permet de préciser que les jetons passés sont à chercher dans les blocs délimités par un espace ou une virgule.
"%a %b %c" : permet de dénombrer le nombre de valeurs à afficher.
À titre de test saisissez cette commande :
for /f "eol=- tokens=1,2 delims=, " %a in (fichier.txt) do @echo %a %b %c %d
Les deux dernières variables seront indiquées comme étant non "remplies" puisqu'un nombre insuffisant de jetons a été alloué.
L'affichage des deux dernières colonnes s'obtient donc en saisissant :
for /f "eol=- tokens=3,4 delims=, " %a in (fichier.txt) do @echo %a %b
Nous pouvons résumer la commande en disant : "Pour chaque ligne analysée, passe le troisième et le quatrième jeton jusqu'à ce que soient renseignées les variables %a et %b qui correspondront aux blocs de données n°3 et 4".

Récupérer différentes variables à partir d'un fichier
Dans un nouveau document nommé Fichier.txt copiez ceci :
C:\Sauvegarde
C:\Test
Dans un nouveau fichier Batch copiez ceci :

```
@echo off
set Fichier=%1
set Dossier=%2
for /f %%a in (fichier.txt) do call :Commande %%a
:Commande
set source=%1
copy %source%%fichier% %dossier%
```

Ce Batch se lance de cette façon : Nom_Batch *.txt c:\temp

Nous récupérons les deux paramètres saisis par l'utilisateur (*.txt et c:\temp) et définissant deux variables (Fichier et Dossier) qui correspondent respectivement à %1 et %2.
Comme nous l'avons dit précédemment le signe % doit être redoublé quand il est utilisé dans un fichier Batch.
À chaque ligne trouvée dans Fichier.txt nous appelons les commandes inclues dans l'étiquette :Commande.
Le premier paramètre trouvé %1 (c:\sauvegarde) est passé en tant que variable nommée %source%.
La ligne suivante se contente d'effectuer le processus de copie en concaténant les variables de telle sorte que la commande s'exécute.

Analyser une chaîne de caractères
Il vous est possible d'analyser le contenu d'un fichier d'une commande ou d'une variable d'environnement en les plaçant entre deux guillemets simples et inversés :
for /f "usebackq" %a in (`set`) do @echo %a
Le commutateur /f vous permet de spécifier plusieurs jetons d'analyse.
Dans cet exemple, l'utilisation du mot clé Usebackq est nécessaire puisque Jeu est placé entre accents inversés. Si vous ne souhaitez n'avoir que les noms sans les chemins correspondants, saisissez :
for /f "usebackq delims==" %a in (`set`) do @echo %a
Le principe est de spécifier un jeu de séparateur.
Dans notre exemple, la ligne "windir=C:\WINDOWS" sera coupée à partir du signe égal. Il ne s'affichera plus que "windir".
Il vous tout à fait possible de faire la même opération sur un fichier nommé Fichier.txt :
for /f "usebackq delims=" %a in (`Fichier.txt`) do @echo %a
Le Bloc-notes s'ouvrira directement dans le fichier spécifié puisque qu'il lui est demandé d'être exécuté...
Curieusement, le mot-clé "usebackq" n'est jamais employé ! Comme nous le verrons dans les paragraphes suivants, il est en effet plus simple de saisir, par exemple, ces commandes :
for /f "delims==" %a in ('set') do @echo %a
for /f "delims=" %a in (Fichier.txt) do @echo %a

Utiliser un filtre avec la commande For
La commande "For" ne s'accommode pas d'un filtre de redirection. Dans l'exemple suivant le symbole de redirection | n'a aucun effet :
for %%a in (*.txt) do type %%a | more
Le symbole | étant repris pour le compte de la commande "For" et non de "Type". Vous devez en passer par la création de deux scripts : L'un contiendra ceci :
for %%a in (*.txt) do test %%a

Nous pouvons aussi écrire : for %%a in (*.txt) do call test %%a
Ou encore : for %%a in (*.txt) do command /c test %%a
Dans l'autre fichier (appelé dans notre exemple Test) copiez ceci :
@type %1 | more
Les fichiers seront triés en fonction de leur nom.

Récupérer le résultat d'une commande en tant que variable
Imaginons que nous souhaitons récupérer le nom du dernier fichier sans son extension et de nous en servir comme d'une variable, nous pouvons nous inspirer de ce script :

```
@echo off
for /f %%a in ('dir /b /od /a-d *.*') do set variable=%%~na
echo Le nom du dernier fichier est : %variable%
```

Le principe du script consiste à récupérer la dernière ligne affichée par la commande "Dir" et de transformer cette ligne en une variable. Pour ce faire, seule le nom du fichier seul est extrait (%~na).

Comparer si deux fichiers font la même taille
Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set fichier1=%1
set fichier2=%2
for %%a in (%fichier1%) do set taille1=%%~za
for %%a in (%fichier2%) do set taille2=%%~za
if %taille1%==%taille2% (
echo Les tailles des fichiers sont identiques.
) else (
echo Les tailles des fichiers ne sont pas identiques.
)
```

Deux variables récupèrent les deux paramètres saisis par l'utilisateur. Ce sont les deux noms de fichiers à comparer.
Pour chaque fichier, nous procédons à l'extraction de la taille (%~za) et créons à chaque fois une variable.
Si les deux tailles sont identiques nous affichons un message de confirmation sinon nous signalons que les tailles trouvées ne sont pas identiques.

Assurer le traitement conditionnel des commandes
Prenons un exemple :
if not exist Fichier.txt echo "Le fichier est manquant"
Nous pouvons également imaginer ce type de commande :

if exist Fichier.txt (del fichier.txt) else echo "Le fichier est manquant"
En bref, nous demandons simplement que si le fichier Fichier.txt est présent dans le répertoire par défaut il faut le supprimer sinon le message suivant apparaîtra : "Le fichier est manquant". La syntaxe de la commande est variable :
if [not] errorlevel Nombre Commande [else Expression]
if [not] Chaîne1==Chaîne2 Commande [else Expression]
Cela suppose que Chaîne1 doit être équivalent à Chaîne2
if [not] exist Nom_Fichier Commande [else Expression]
Si les extensions de commandes sont activées, il est possible d'utiliser la syntaxe suivante :
if [/i] Chaîne1 Options_Comparaison Chaîne2 Commande [else Expression]
Le commutateur /i permet de ne pas tenir compte de la casse.
Le tableau suivant récapitulatif les valeurs possibles pour Options_Comparaison :

Opérateur Description
EQU Égal à
NEQ Différent de
LSS Inférieur à
LEQ Inférieur ou égal à
GTR Supérieur à
GEQ Supérieur ou égal à

Examinons cette commande : if cmdextversion Nombre Commande [else Expression]
Le mot clé cmdextversion permet de vérifier si les extensions de commandes sont activées ou non. La première version est égale à 1. Elle doit être indiquée dans "Nombre".
Voici une autre possibilité : if defined Variable Commande [else Expression]
Ne reconnaît la condition comme étant vraie que si "Variable" est défini.
Astuce : Un cas particulier concerne la vérification des répertoires. Si nous souhaitons nous assurer que le répertoire Test existe bien nous devrons saisir : if exist Test\nul (@echo "Le répertoire existe bien !") else md Test
Cela revient à dire que si le répertoire Test existe bien un périphérique fictif (nul) existe aussi.
Nous pouvons imaginer la création d'un fichier de commande qui se chargera de vérifier si un répertoire Test existe sur le disque dur. Si la commande précédente échoue (errorlevel 0) alors il sera créé. Copiez dans un nouveau fichier Batch ces trois lignes :

```
echo off
dir /s Test
if errorlevel 0 (md Test)
```

Nous pourrions écrire la dernière ligne de cette façon :
if %errorlevel% leq 0 (md c:\Test)
Dans ce dernier cas nous nous servons de errorlevel comme d'une variable qui sera égale ou non à 0.
Nous pouvons aussi saisir ce type de commande :
if not exist c:\Test\nul md c:\Test
Vous pouvez conditionner la recherche à la condition que l'utilisateur saisisse le nom du fichier à rechercher :

```
@echo off
if not exist %1 (echo Le fichier %1 est introuvable dans ce répertoire.) else echo Fichier trouvé !
```

La variable %1 représente dans ce cas la chaîne de caractères (le nom du fichier) que l'utilisateur aura saisie à la suite du nom du fichier de commande.

Créer un contrôle de saisie
Dans le cas d'un fichier Batch nécessitant la saisie d'un paramètre vous pouvez insérer en début de script une routine de contrôle. Vous pouvez tester ces deux méthodes en saisissant ou non n'importe quelle variable en début de votre fichier batch :

```
if [%1] ==[] echo Paramètres de commande manquants
if [%1] equ [] echo Paramètres de commande manquants
```

Cela revient à dire : si aucune chaîne de caractères n'a été saisie à la suite du nom du fichier de commandes affiche ce message : "Paramètres de commande manquants". Voici un autre exemple :

```
@echo off
if {%1}=={} @echo Saisissez le nom du fichier&goto :eof
if not exist %1 (@echo %1 n'existe pas) else @echo Le fichier existe !
```

La première ligne s'assure qu'à la variable %1 correspond une saisie quelconque. Si le paramètre qui suit le nom de la commande {%1} est égale == à rien {} ou, plus exactement, si rien n'a été saisi au clavier alors un message d'avertissement sera affiché.
Si le paramètre saisi ne correspond (dans ce cas) à aucun fichier existant alors affiche cet autre message d'avertissement.
L'étiquette :goto:eof permet de quitter le fichier de commande si la condition posée est vraie. Si la condition est fausse, la commande suivante sera exécutée. Cela évite d'insérer une étiquette :fin à la fin de notre script.

Il y a une méthode plus simple : inspirez-vous de ce script :

```
@echo off
if not %1[==[ if exist %1 goto message
echo Le fichier n'existe pas ou rien n'a pas été précisé & goto:eof
: message
echo Bonjour !
```

De cette façon, nous vérifions en une seule commande que l'utilisateur a bien saisi un paramètre et que le fichier spécifié est bien présent.

Choisir à quel étage l'ascenseur va s'arrêter
Dans un nouveau Batch copiez ce contenu :

```
@echo off
if "%1" == "3" goto niveau3
if "%1" == "2" goto niveau2
if "%1" == "1" goto niveau1
if "%1" == "" goto :eof
:niveau3
CD ..\..\..\
goto :eof
:niveau2
CD ..\..\
goto :eof
:niveau1
CD ..\
goto :eof
```

Copiez ce script dans un des chemins définis par la variable d'environnement %Path%. Saisissez à partir de n'importe quelle arborescence le nom du Batch suivi du numéro de niveau auquel vous souhaitez remonter.
Si, par exemple, nous saisissons le chiffre 3 nous exécuterons cette commande : CD ..\..\..\.

Créer une boucle
Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set variable=
: Boucle
set /a variable+=1
if /i %variable% equ 10 goto :eof
echo %variable% && goto Boucle
```

Nous posons une variable sans la définir. Pour vous en rendre compte saisissez ces commandes :
set variable=
set variable
Nous créons une étiquette nommée :Boucle qui va effectuer les actions suivantes :
set /a variable+=1 : incrémenter la variable de départ d'un pas de 1. De 0 notre variable va donc être égale à 1.
if /i %variable% equ 11 goto :eof : Si la variable est égale au nombre 10 arrêter l'exécution du script.
Dans le cas contraire inscrire à l'écran la valeur obtenue puis reprendre l'exécution du fichier de commandes à partir de l'étiquette nommée :Boucle.
Si nous voulons obtenir un pas de "deux" il suffit de modifier la ligne set /a variable+=1 par ceci : set /a variable+=2

Créer un compteur
Créez un fichier Batch avec ce contenu :

```
@echo off
set filtre=*.*
set taille=0
set compteur=0
for /r %%a in (%filtre%) do if %%~za==%taille% (
set /a compteur+=1
)
echo Il y a %compteur% fichiers dont la taille est de 0 octet.
```

À l'aide de la commande "Set" nous définissons trois variables :
filtre : dans ce cas la recherche portera sur tous les fichiers *.*.
taille : définit la taille du fichier à chercher. Dans notre cas : 0 ko.
compteur : qui est tout d'abord définit à 0 puis incrémenté à chaque fois de 1 (set /a compteur+=1) dès qu'un nouveau fichier répondant aux critères définis a été signalé.
La commande for /r démarre une recherche récursive jusqu'à ce que plus aucun nouveau fichier ne soit trouvé.
Chaque nom de fichier est passé sous forme de variable (%a) qui se réinitialise à chaque nouveau fichier trouvé et le lot de commandes exécuté.
Le commutateur %~za développe %a jusqu'à la taille du fichier.
L'utilisation des parenthèses nous permet de définir que si les critères de recherche sont remplis alors la variable %compteur% est incrémenté d'une unité.
Le commutateur /a permet d'indiquer que c'est une expression numérique qui sera évaluée pour la variable %compteur%

La dernière ligne affiche simplement la dernière valeur de la variable %compteur%