Les fichiers de commande

Fichiers ".bat" ou ".cmd"

Un fichier de commande est un fichier texte contenant des commandes en mode invite de commande qui seront lues par l'interpréteur de commande COMMAND.COM ou CMD.EXE. COMMAND.COM est l'interpréteur que l'on connaît depuis les débuts du DOS, il est devenu CDM.EXE depuis Windows NT.
Un fichier de commande peut avoir n'importe quel nom excepté celui d'une commande existante. L'extension est généralement ˝.bat˝ mais depuis Windows NT l'extension ˝.cmd˝ est permise aussi.

On utilise les fichiers de commande pour créer des procédures automatiques pouvant être lancées par des personnes qui en ignorent les détails ou pour automatiser des séquences fréquemment utilisées.

Un fichier de commande contient : une suite de lignes de commandes initialement destinées à être saisies au clavier + des instructions propres aux algorithmes (affichages, tests, sauts, lectures et modifications de variables, etc.)

Les commandes utilisées en saisie directe sont supposées connues. Nous allons passer en revue celles qui ne prennent tout leur sens que dans le cadre des fichiers de commande pour dialoguer avec l'utilisateur ou pour manipuler des variables.

ECHO

La commande Echo peut vous rendre deux types de service:

1° MS-DOS affiche généralement chaque ligne du fichier de commande au moment où il l'interprète, juste avant de l'exécuter. Cet "écho" des commandes peut être désactivé par la commande Echo Off ou autorisé à nouveau par la commande Echo On.

Echo Off	Plus d'écho sur la sortie standard (l'écran)
Echo On	L'écho est à nouveau autorisé
Echo	Selon que l'écho des commande est autorisé ou non affiche : Commande ECHO activée ou Commande ECHO désactivée

2° Cette même commande est aussi utile pour afficher des messages à l'écran.

Echo *%variable%*	Affiche la valeur de la variable
Echo *Chaîne*	Affiche simplement cette chaîne de caractères
Echo.	Echo suivi d'un point provoque un retour à la ligne

La commande Echo est inutile en mode de saisie directe mais il est tout de même possible de la tester dans ce mode. Si vous tapez la commande Echo Off vous risquez de ne plus voir apparaître l'invite de commande, faites Echo On pour retrouver vos marques.

@

Le @ placé en début de ligne dans un fichier de commande empêche l'écho de cette ligne sans pourtant en interdire l'exécution.

Le fichier de commande débute souvent par @Echo Off. Le @ masque l'affichage de cette ligne de commande. La suite de l'instruction, la commande Echo Off empêche les échos des commandes suivantes.

REM

La commande Rem, placée en début de ligne, a pour fonction d'introduire un commentaire dans le code du fichier de commande. Elle est aussi très pratique pour désactiver une ligne de commande.

PAUSE

La commande Pause interrompt l'exécution du fichier de commande en affichant :
Appuyer sur une touche pour continuer...
Si ce message ne vous convient pas, annulez-le en le redirigeant vers la sortie fictive NUL. (PAUSE > NUL)

BREAK

La commande Break [On | Off] active ou désactive le contrôle étendu du CTRL+C ou CTRL+BREAK.
Normalement, MS-DOS ne teste la saisie des touches CTRL+C que lorsqu'il utilise les fonctions du clavier et de l'affichage à l'écran ou sur l'imprimante.
Suite à la commande Break On, ce test des touches CTRL+C est fait plus souvent, notamment chaque fois que le programme fait un accès au disque.
La commande Break utilisée sans argument permet de connaître l'état de cette commande.

Les variables

Les variables reconnues par l'interpréteur du langage de commande peuvent avoir plusieurs origines:
- des variables d'environnement ; exemple: echo %username%
- les paramètres passés au fichier de commande %1, %2, … %9
- le nom du fichier de commande lui-même : %0
- une nouvelle variable SET N=10
- la variable ERRORLEVEL, elle contient la valeur rendue par le dernier programme exécuté. En principe, un programme bien conçu rend 0 s'il a pu s'exécuter sans erreur.
Quand une erreur a eu lieu, le code rendu par le programme doit pouvoir servir à déterminer le type d'erreur qu'il a constaté.

Variables d'environnement

Les variables d'environnement sont des données auxquelles le système d'exploitation réserve une partie de la mémoire vive. Elles sont accessibles à n'importe quel programme. Parmi eux, l'interpréteur de commandes recherche le contenu de la variable PATH pour connaître les emplacements des exécutables.
Ces variables sont créées en leur fixant une valeur par la commande SET :

```
SET Machin=Skoubidou↵
```

La valeur d'une variable s'obtient en entourant son nom avec des signes %

```
Echo %Machin%↵
```

On supprime une variable avec la commande SET sans rien derrière le signe égal.

```
SET Machin=↵
```

Exemple de modification :

```
PATH=%PATH% ;\Util↵
```

Exemple d'utilisation :

```
DIR %TEMP%↵
```

Certains logiciels utilisent des variables d'environnement qui leur sont propres.

La libraire standard du C possède des fonctions getenv() et putenv() pour ce genre d'opérations.

Exemple :

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char* argv[])
{
      char *valeur;
      valeur = getenv("OS");
      printf("La variable OS contient %s\n", valeur);

      putenv("Zigoto=Tartempion");
      valeur = getenv("Zigoto");
      printf("La variable Zigoto contient %s\n", valeur);
      return 0;
}
```

→ Résultat :

```
La variable OS contient Windows_NT
La variable Zigoto contient Tartempion
```

SET

SET [variable=[chaîne]]

`SET↵`	Affiche toutes les variables d'environnement
`SET a↵`	Affiche toutes les variables dont le nom commence par a
`SET variable=chaîne↵`	Donne à la variable la valeur chaîne
`SET variable= ↵`	Supprime la variable

Attention, les espaces autour du signe égal comptent.

```
SET A = abc↵
```

Le nom de la variable comporte deux caractères : le A suivi d'un espace.
La valeur de la variable en compte 4 : un espace suivi de abc.

SET /A [variable=]"expression"

L'option "Assignation" est disponible depuis les extensions de commandes apparues depuis Windows NT. Cette option /A permet d'évaluer une expression arithmétique ou logique et d'en attribuer la valeur à une variable. Nous plaçons les expressions entre guillemets. Un nom de variable dans une expression représente sa valeur. Il n'est donc plus nécessaire de placer les noms entre % .

Les valeurs numériques sont décimales (base 10) par défaut. On les fait précéder du préfixe 0x pour les valeurs hexadécimales (base 16) ou par 0 si elles sont saisies en octal (base 8)

Exemples :	SET /A "2+3"	donne 5	(car 2+3= 5)
	SET /A "0xB+6"	donne 17	(car 0xB = 11 et 11+6=17)
	SET /a "14 & 3"	donne 2	(car 1110 AND 0011 = 0010)

SET /P variable="invite"

SET avec l'option "Prompt" attribue à la variable une valeur saisie par l'utilisateur. La commande commence par afficher l'invite avant de lire l'entrée.

Les sauts de programmes

Les fichiers de commande ne sont pas toujours de simples séquences d'instructions. Il arrive dans certaines conditions que des instructions doivent être ignorées ou au contraire être répétées un certain nombre de fois. Cela est possible en combinant les instructions de tests et les instructions de sauts.

:label

Le label est un nom quelconque précédé de deux points. Il marque un endroit donné du code pour servir de destination à la commande GOTO

GOTO

La commande GOTO suivie d'un label provoque un saut dans le programme à l'endroit où est défini le label.

Exemple : GOTO label

La commande GOTO :EOF permet de sortir d'un fichier de commande. On se sert de cette forme de la commande GOTO pour faire l'équivalent d'un « return » en fin d'une sous-routine.

CALL

La commande CALL a été prévue initialement pour appeler un fichier de commande à partir d'une autre. Une fois que les commandes du fichier appelé sont achevées on revient à l'instruction qui suit le call.
Exemple : CALL CmdFile2 Arg1 Arg2 appelle le fichier de commande CmdFile2 en lui passant les arguments Arg1 et Arg2. Les commandes du fichier appelé étant achevées on revient à l'instruction qui suit le call.

Ce retour à l'instruction suivante n'aurait pas eu lieu si on avait appelé CmdFile2 en inscrivant son nom comme une commande sans utiliser la commande Call.

Depuis Windows NT, (les extensions de commande), la commande Call accepte les étiquettes comme cible. Pensez à terminer la routine visée par la commande GOTO :EOF pour assurer le retour sous la commande qui a fait l'appel.

L'alternative

La syntaxe de la commande IF dépend de la version du système d'exploitation. Le mot ELSE fait partie des extensions de commandes dont on ne dispose que depuis Windows NT. Ainsi avec Windows95 ou 98 le mot ELSE n'est pas pris en compte. Pour connaître la syntaxe admise par votre système d'exploitation demander son aide en tapant la commande "IF /? ".

NB. Les mots clés, dans les versions très anciennes du DOS, doivent s'écrire en majuscules.

IF *Condition Commande*

Soumet l'exécution de la commande à une condition. Si la condition est fausse la commande est ignorée

IF *Condition Commande_1* ELSE *Commande_2*

Suivant que la condition est vraie ou fausse, ce sera la Commande_1 ou la Commande_2 qui s'exécutera.

IF est une commande qui comme toutes les autres se termine par un retour à la ligne.
En principe donc, les instructions IF et ELSE doivent être écrites sur une seule ligne. Ce n'est pas toujours possible car certaines commandes du DOS doivent elles-aussi s'achever par un retour à la ligne. On parvient à concilier ces deux exigences en plaçant les commandes entre parenthèses.

Exemples :

IF *Condition* (*Commande_1*) ELSE *Commande_2*	Ici les commandes sont bien terminées par des retours à la ligne

IF *Condition* (
  Commande_1
) ELSE (
  Command_2
)
	Commande_1 et commande_2 peuvent être remplacées par des blocs de plusieurs lignes de commandes.
	Cette disposition fait penser à celle du langage C ou du Java.

Malgré l'artifice des parenthèses, la commande IF reste à considérer comme une commande en une seule ligne. Cela a une conséquence étrange, qu'il faut savoir prévoir : La ligne est lue complètement en échangeant les variables mises entre % avec leur valeur avant l'exécution. Ce remplacement est appelé « l'expansion » des variables. Il est fait à la lecture et non pas lors de l'exécution de la ligne de commande.

Ainsi la « ligne » suivante

```
set VAR=avant
if "%var%" == "avant" (
  set VAR=après
  ECHO VAR
)
```

… affiche ceci !

```
C:\>set VAR=avant
C:\>if "avant" == "avant" (
 set VAR=après
 ECHO avant
)
avant
C:\>
```

MS-DOS interprète une commande entre parenthèses comme si elle était terminée par un retour à la ligne.
IF *Condition* **(** *Commande_1* **)** ELSE *Commande_2*

Expressions des conditions

NOT *condition*	Si la condition est fausse
ERROLEVEL nombre	Si le code rendu par le programme précédent est supérieur ou égal au nombre indiqué
DEFINED variable	Si la variable est définie
EXIST fichier	Si ce fichier existe
chaîne1 == chaîne 2	Si chaine1 est identique à chaîne2

D'autres opérateurs de comparaisons sont aussi fournis par les extensions de commandes dont sont dotés les interpréteurs de commandes les plus récents (CMD.exe)

chaîne1 EQU *chaîne2*	Si chaîne1 est identique à chaîne2
chaîne1 NEQ *chaîne2*	Si chaîne1 est différente de chaîne2
chaîne1 LSS *chaîne2*	Si chaîne1 est plus petite que chaîne2
chaîne1 LEQ *chaîne2*	Si chaîne1 est inférieur ou égale à chaîne2
chaîne1 GTR *chaîne2*	Si chaîne1 est supérieur à chaîne2
chaîne1 GEQ *chaîne2*	Si chaîne1 est supérieur ou égale à chaîne2

L'option /I demande d'ignorer la casse lors de ces comparaisons. Si les deux chaînes ne contiennent que des chiffres, alors la comparaison est numérique au lieu d'être alphabétique.