document autorisé pour l'évaluation finale du module
NE RIEN ECRIRE SUR CE DOCUMENT

Les fichiers de commandes

I] Bases

➢ Définition

Un **fichier de commandes**, encore appelé **script** ou fichier **batch**, contient une suite de commandes qui est exécutée automatiquement. Chaque ligne du fichier est exécutée individuellement. On parle aussi de "traitement par lot".

➢ Création / Modification

Pour créer un fichier de commandes, on utilise un éditeur de texte (`Bloc-Notes, WordPad, …`) et on l'enregistre avec l'extension .cmd ou .bat avec le `Type Tous les fichiers, Documents texte MS-DOS, ….`
Pour modifier un fichier de commandes, clic droit sur le nom du fichier et sélectionner `Modifier`.

➢ Exécution

L'exécution d'un fichier de commandes peut se faire soit :
- directement depuis l'interface du système par double clic sur le nom du fichier ;
- dans une fenêtre d'exécution MS-DOS, en saisissant le nom du fichier.

Pour ouvrir une fenêtre d'exécution MS-DOS, il faut soit, dans `Démarrer→Tous les programmes→Accessoires` :
- `→Exécuter`, taper la commande `cmd` ;
- double cliquer sur `Invite de commandes`.

➢ Mécanisme de bases

`ECHO OFF`	pour ne pas afficher les commandes du fichier de commandes
`@ECHO OFF`	pour ne pas afficher les commandes, commande `ECHO OFF` comprise
`PAUSE`	invite l'utilisateur à appuyer sur `<Entrée>` pour continuer
`REM` ou **`::`**	pour insérer un commentaire
`( )`	permet de regrouper des commandes (une par ligne)
`&`	pour enchaîner des commandes sur une même ligne
`>` et `>>`	pour rediriger la sortie vers un fichier (par exemple : `DIR > liste.txt`)
`^`	annule l'interprétation du caractère spécial qui suit (par exemple : `^>`)
`:etiq`	pour définir une étiquette de nom `etiq` (à utiliser avec `GOTO`)
`GOTO` `etiq`	effectue un saut à l'étiquette spécifiée (ici `etiq`); **`GOTO :EOF`** pour aller à la fin du fichier (End Of File)
`%variable%`	désigne le contenu de la variable de nom `variable`
`2`	sortie des erreurs d'une commande `2>NULL` pour diriger la sortie des erreurs vers la poubelle
|	redirige la sortie d'une commande vers le filtre `more` pour une présentation page par page ou le filtre `SORT` pour trier la sortie (par exemple : `DIR | MORE`)

- **Variables système**

Un fichier de commandes peut accéder aux variables d'environnement mais aussi aux variables suivantes :

`CD`	le répertoire courant
`DATE`	la date
`TIME`	l'heure
`RANDOM`	un entier tiré aléatoirement entre 0 et 32767_{10}
`ERRORLEVEL`	le code retour de la dernière commande
`CMDCMDLINE`	ligne de commande originelle invoquée par le processeur de commande

- **Variables utilisateur**

`ECHO %x%`	affiche le contenu de la variable `x`
`ECHO %x:~i,j%`	affiche le contenu de la variable `x` depuis la position `i` et pour `j` caractères la première position étant au rang `0` (`i` et `j` étant des entiers) *remarque*: on peut aussi utiliser ce format avec la commande `SET`
`ECHO %x:~i%`	affiche le contenu de la variable `x` depuis la position `i` jusqu'à la fin *remarque*: on peut aussi utiliser ce format avec la commande `SET`
`SET x=val`	crée la variable `x` puis l'initialise avec la valeur `val`
`SET car`	affiche toutes les variables commençant par le caractère `car`
`SET /P x=mesg`	affiche à l'écran `mesg` puis affecte à `x` la valeur saisie au clavier
`SET /A calcul`	effectue l'opération arithmétique `op` entre les deux variables `var1` et `var2` si `calcul` est exprimé par `var=%var1%op%var2%` et met le résultat dans `var`; les opérateurs `op` autorisés sont `+ - / * % << >> & | ^` `calcul` peut aussi être exprimé par `var op=valeur` ou encore par `var op=%var1%` (par exemple : `SET /A n+=1` ou encore `SET /A x+=%y%`)
`SET x=%x:s1=s2%`	remplace dans la variable `x` la chaîne `s1` par la chaîne `s2` *remarque*: on peut aussi utiliser ce format avec la commande `ECHO`

bienvenue.bat

```
@ECHO OFF
SET /P x=Entrer votre prenom :
ECHO Bienvenue %x% a l'IUT Info Lyon
PAUSE
```

II] Structure de contrôle conditionnelle

- Il existe plusieurs formes pour écrire une structure de contrôle conditionnelle.

1°) Exécuter la commande `cde` si la condition `c` est vraie.
attention : tout doit tenir sur une seule ligne

```
IF   c   cde
```

2°) Pour exécuter plusieurs commandes, on peut les enchaîner séquentiellement avec le caractère &.

```
IF   c   cde1   &   cde2   &   cde3
```

3°) On peut regrouper les commandes dans un bloc commençant par `(` et finissant par `)`.

```
IF   c   (
       cde1
       cde2
       cde3
)
```

4°) Il est possible d'exécuter des commandes placées à une étiquette (par exemple `suite` ici).

```
IF   c   GOTO suite
```

- La condition `c` peut s'écrire de plusieurs façons.

`%var%==valeur`	vrai si le contenu de la variable `var` est égale à `valeur`
`%var1%==%var2%`	vrai si le contenu de la variable `var1` est égale à la variable `var2`
`NOT %var%==valeur`	vrai si le contenu de la variable `var` n'est pas égale à `valeur`
`NOT %var1%==%var2%`	vrai si le contenu de la variable `var1` n'est pas égale à la variable `var2`
`EXIST %fic%`	vrai si le contenu de la variable `fic` est un nom de fichier qui existe
`NOT EXIST %fic%`	vrai si le contenu de la variable `fic` est un nom de fichier qui n'existe pas

remarque: si la condition `c` est le résultat d'une commande, comme `EXIST` par exemple, la structure de contrôle `IF` peut s'écrire avec un `ELSE` à condition qu'elle soit sous la forme :

```
IF    c    (
           cde1
           cde2
)   ELSE   (
           cde3
)
```

condition.bat

```
@ECHO OFF
SET fic=essai
IF   EXIST %fic%   (
            TYPE %fic%
)   ELSE   (
            ECHO Le fichier %fic% n'existe pas.
)
```

Pour comparer des variables entre elles, il existe aussi la forme :

```
IF /I   %var1% op_comp %var2%   ...
```

ou pour comparer une variable avec une valeur :

```
IF /I   %var1% op_comp valeur   ...
```

L'opérateur de comparaison `op_comp` peut être un des six opérateurs suivants :

`EQU`	égal à
`NEQ`	différent de
`LSS`	inférieur à
`LEQ`	inférieur ou égal à
`GTR`	supérieur à
`GEQ`	supérieur ou égal à

exemple

```
IF   /I   %ERRORLEVEL% NEQ 0   GOTO debut
```

III] Arguments passés à un fichier de commandes

Il est possible de passer des arguments à un fichier de commandes. Ceci se fait sur la ligne de commande, au moment de l'appel du fichier.

```
>nom_fichier_commandes argument_1 argument_2 argument_3 …
```

- Les arguments sont récupérés dans le fichier de commandes par leur position sur la ligne de commande (`1` pour le premier, `2` pour le second, ...). Le nom de la commande est donné à la position `0`.

test_arg.bat

```
@ECHO OFF
ECHO %0 est le nom de la commande
ECHO %1 est le premier argument
ECHO %2 est le second argument
PAUSE
```

Dans une fenêtre d'exécution MS-DOS, placez-vous dans le dossier contenant le fichier test_arg.bat puis taper :

```
> test_arg abc 12345
```

Vous devez obtenir :

```
test_arg est le nom de la commande
abc est le premier argument
12345 est le second argument
Appuyez sur une touche pour continuer …
```

- La commande `SHIFT` décale d'un cran à gauche les arguments en écrasant le premier par le second et ainsi de suite, donc une boucle permet de récupérer et de traiter un à un les arguments passés sur la ligne de commande.

compte_arg.bat

```
:: fichier de commandes qui affiche et compte les
:: arguments passés en ligne de commande
@ECHO OFF
SET n=0
:copie
IF   "%1"==""    GOTO fin
SET /A n+=1
ECHO argument %n% = %1
SHIFT
GOTO copie

:fin
ECHO nombre d'argument(s) lu(s) = %n%
PAUSE
```

IV] Appeler un fichier de commandes depuis un autre fichier de commandes

La commande `CALL` permet d'appeler :

- soit une étiquette avec la syntaxe `CALL :etiq` et, dans ce cas, les instructions contenues après l'étiquette `etiq` sont exécutées ;
- soit un autre fichier de commandes avec la syntaxe `CALL fichier` et la possibilité de passer des arguments.

Dans les deux cas, le déroulement du fichier de commandes "appelant" n'est pas stoppé et reprend là où on l'a laissé une fois "l'appelé" exécuté.

appel.bat

```
:: fichier de commandes qui appelle un autre fichier de commandes
@ECHO OFF
SET /P x=Saisir un entier :
SET /P y=Saisir un entier non nul :
IF   /I %y% NEQ 0   (
      CALL division %x% %y%
)   ELSE   (
      ECHO Le dernier entier saisi est nul.
)
PAUSE
```

division.bat

```
SET /A n=%1/%2
ECHO %1/%2=%n%
```

V] Structure de contrôle répétitive

- Il existe plusieurs formes pour écrire une structure de contrôle répétitive.

1°) Répéter la commande `cde` pour toutes les valeurs possibles de `var` prises parmi `ensemble`.
attention : tout est sur une seule ligne et la variable est désignée par `%%var`

```
FOR %%var IN (ensemble) DO cde
```

2°) Répéter la commande `cde` pour toutes les valeurs possibles de `var` prises depuis `debut` jusqu'à `fin` avec incrémentation d'une valeur égale à `pas`.

```
FOR /L %%var IN (debut,pas,fin) DO cde
```

boucles.bat

```
@ECHO OFF
FOR %%i IN (1 3 5 7 9) DO ECHO %%i
REM ------------------------------ structures identiques
FOR /L %%j IN (1,2,9) DO ECHO %%j
PAUSE
```

3°) Plusieurs commandes peuvent être exécutées dans la boucle si elles sont enchaînées avec le caractère & sur la ligne du `FOR`.
4°) Une commande est exécutée récursivement avec l'option `/R` dans le `FOR`.

fichiers.bat

```
:: fichier de commandes qui affiche tous les fichiers du
:: répertoire courant et de ses répertoires
@ECHO OFF
FOR /R %%a IN (*.*) DO ECHO %%a
PAUSE
```

5°) Lorsque le nombre de commandes à exécuter dans la boucle est trop important pour tenir sur une seule ligne :

- on place les commandes à une étiquette (`:affichage` dans le fichier tirage.bat) ;
- on appelle sur la ligne du `FOR` cette étiquette par `CALL :affichage %%var` si `var` est la variable de boucle ;
- les commandes placées après l'étiquette `:affichage` peuvent récupérer les différentes valeurs de `var` par `%1` (`%%a` est le premier argument de la commande `affichage %%a`).

attention : Il s'agit d'une programmation non structurée.
Toutes les commandes situées après l'étiquette sont exécutées.

Si des commandes sont placées entre le FOR et l'étiquette, c'est tout l'ensemble qui est exécuté ⇨ Il faut penser à forcer la sortie (saut inconditionnel en fin de fichier avec GOTO :EOF).

tirage.bat

```
@ECHO OFF
FOR /L %%a IN (1,1,5) DO CALL :affichage %%a
PAUSE
GOTO :EOF

:affichage
SET k=%1
SET x=%RANDOM%
ECHO Le tirage numero %k% vaut %x%.
```

- Les variables de sortie du FOR peuvent être substituées. Voici quelques possibilités qui peuvent être combinées entre elles.

%~var	%var en supprimant les guillemets
%~fvar	%var en nom de chemin d'accès reconnu
%~dvar	%var en lettre de lecteur uniquement
%~pvar	%var en chemin d'accès uniquement
%~nvar	%var en nom de fichier uniquement
%~xvar	%var en extension de fichier uniquement
%~zvar	%var en taille du fichier
%~dpvar	%var en lettre de lecteur et chemin d'accès uniquement
%~nxI	%var en nom de fichier et extension uniquement

chemin.bat

```
:: fichier de commandes qui affiche uniquement le nom de
:: fichier des fichiers d'extension .bat du répertoire
:: courant
@ECHO OFF
FOR %%a IN (*.bat) DO ECHO %%~na
PAUSE
```

- La commande FOR permet aussi d'analyser le contenu des lignes d'un fichier (ou de la sortie d'une commande).

Chaque ligne est lue et décodée selon la valeur du délimiteur de champs spécifiée par delims et selon les jetons retenus spécifiés par tokens.
Les lignes qui commencent par le caractère spécifié dans eol sont ignorées.
Les différents champs décodés sont récupérés dans un nombre suffisant de variables.

fichier.bat

```
@ECHO OFF
FOR /F "eol=- tokens=1,3,* delims=;" %%a IN (TEST\test.txt) DO ECHO %%a %%b %%c
PAUSE
```

Dans le fichier de commandes ci-dessus, on analyse le fichier test.txt du répertoire TEST. On ignore les lignes qui commencent par le caractère – (eol=-) et, pour les lignes retenues, on affiche le premier champ (tokens=1), le troisième champ (tokens=3) et le reste de la ligne (tokens=,*), les champs étant séparés par le point virgule (delims=;).

VI] Sortie et code retour d'un fichier de commandes

Le code retour d'un fichier de commandes est une valeur entière qui peut être récupérée soit :
- au niveau de l'interpréteur de commande ;
- au niveau du fichier de commandes qui a éventuellement appelé ce fichier de commandes.

Dans les deux cas, c'est la variable système `ERRORLEVEL` qui va permettre de connaître la valeur de cet entier. Le code retour doit être conforme avec le comportement du système pour lequel toute commande qui s'est bien exécutée retourne `0`. Une valeur de `ERRORLEVEL` autre que `0` signale donc une terminaison anormale.

Pour positionner le code retour on utilise soit :
- la commande `EXIT valeur` qui termine le fichier de commandes et la fenêtre d'exécution MS-DOS en cours ;
- soit la commande `EXIT /B valeur` qui termine uniquement le fichier de commandes.

VII] Empiler les fenêtres d'exécution MS-DOS

Lorsqu'une fenêtre d'exécution MS-DOS est ouverte, on peut :
- empiler des processus `CMD` en tapant la commande `CMD` ;
- dépiler des processus par la commande `EXIT` qui termine le dernier processus `CMD.`

Il est donc possible de créer un nouveau processus `CMD` avant de lancer un fichier de commandes. Ceci permet de s'assurer que la fenêtre d'exécution MS-DOS mère ne sera pas fermée si le fichier de commandes fait un `EXIT` sans l'option `/B`, puisque dans ce cas là, c'est le processus `CMD` fils qui se terminera.