

Enseignant responsable : Amir Souissi

TP N° 3

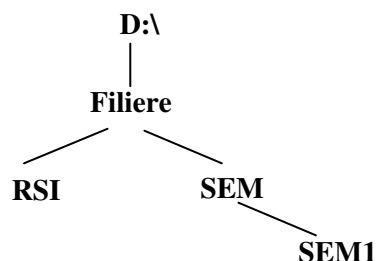
Atelier Système 1

1) Les fichiers de commandes

- Les fichiers de commandes ou fichiers *batch* permettent le traitement automatique des tâches répétitives. L'extension d'un fichier *batch* est « **.bat** ».
- Ils se présentent sous forme de *fichiers textes* contenant une ou plusieurs *commandes MS-DOS*.
- Chaque ligne est lue séquentiellement, donc chaque commande du fichier est exécutée au fur et à mesure. Lorsque toutes ces commandes sont épuisées, le système d'exploitation reprend la main. On parle alors de « *traitement par lots* ».
- Pour les lancer, il suffit de saisir leurs noms. On peut aussi les exécuter dans l'environnement graphique (par un double clic).
- Les fichiers batch sont *emboîtables*.
- L'interruption de l'exécution d'un fichier batch est possible par **ctrl+c**

1.1.Exemples de fichier batch

- Soit le fichier batch nommé « Essai.bat » :
Echo ceci est mon premier batch
*Dir *.txt*
*Dir *.bat*
Time
Date
- Pour exécuter ce fichier il suffit de taper son nom dans l'invite de commandes: Essai
 - ❖ **La commande ECHO :**
- MS-DOS affiche sur l'écran les commandes des fichiers batch avant de les exécuter. L'instruction **ECHO** dirige cette visualisation.
 - ECHO OFF inhibe cette visualisation
 - ECHO ON rétablit cette visualisation (mode normal)
 - ECHO sans argument, délivre à l'écran le mode actuel (on ou off)
 - ECHO mesg dirige le contenu du message mesg vers l'écran
- **Exécuter le premier exemple de fichier batch avec les deux modes (on et off) de echo.**
L'utilisation de l'@ permet de ne pas afficher le statut de la commande ECHO
Exemple : **@echo off**
- **Tester @echo off au début du fichier précédent et remarquez les différences ?**
- **Créer un fichier portant le nom Bienvenue et d'extension .bat qui doit effacer l'écran et affiche le message "Voici mon premier script DOS" ? Utiliser la commande echo qui affiche à l'écran la chaîne passée en paramètre.**
- **Ecrire et tester un fichier .bat qui porte le nom arbo et qui crée l'arborescence de dossiers suivante puis affiche l'arborescence de Filiere.**



1.2. Les variables

- Certaines variables sont prédéfinies par Windows (via Autoexec.bat ou par le panneau de configuration). Ce sont des **variables d'environnement** persistantes (c'est-à-dire qu'elles sont toujours utilisables).
 - La commande **SET**, sans paramètre, permet de lister les variables d'environnement.
 - On peut utiliser cette commande pour créer ses propres variables dans le Shell ou dans un fichier batch.
 - Syntaxe : **set variable=valeur (pas d'espace avant le signe =)**
Exemple : SET Mondossier= "C:\Mes Documents\Travail"
 - Pour stocker des nombres et faire des calculs :
Set /a variable=valeur_numérique
Exemple : **Set /a val=20**
 - Pour faire saisir quelque chose à l'utilisateur :
Set /p variable=texte d'invitation :
Exemple : **Set /p age=Veuillez entrer votre age :**
 La saisie sera considérée comme du texte
 Pour faire saisir des nombres on fera :
Set /a age=%age%
- Remarque** : L'utilisation d'une variable se fait en la mettant entre deux %.
- Exemple :
 echo %age% est différent de echo age
 echo %Mondossier%
 echo %val%
- **Ecrire un fichier batch nommé variables.bat qui permet d'initialiser deux variables numériques saisies par l'utilisateur et d'afficher leurs sommes.**
- Pour supprimer une variable, il suffit de la faire suivre du signe = tout seul
Exemple : SET Mondossier=
- **Que fait la commande set p ?**
- **La variable d'environnement PATH** contient tous les répertoires que le système explore si on lui demande d'exécuter un fichier .bat, .exe ou .com *n'existant pas dans le répertoire courant*. La variable PATH est constituée d'une liste de chemins de répertoires séparés par des « ; »
 - Il est possible d'ajouter un chemin dans le path en utilisant la forme suivante :

PATH=%PATH%;C:\MonDossier

Ici, nous avons ajouté C:\MonDossier à la variable Path. Désormais, si nous mettons nos fichiers .bat dans le dossier C:\Mondossier, alors on peut les exécuter de n'importe quel emplacement sans spécifier leurs path.

Remarque : la durée de vie d'une variable déclarée ainsi que la modification d'une variable d'environnement est relative à la session DOS active. C'est-à-dire, à la fermeture de la console les variables déclarés et les modifications effectuées seront perdues.

1.3. Les paramètres

- Les commandes incluses dans le fichier peuvent être paramétrées, c'est à dire que les objets de ces commandes sont des **paramètres formels**.
- Les **paramètres réels** seront fournis lors du lancement du fichier batch.
- Les paramètres formels sont de la forme : **%n**
 0 <= n <= 9
 %0 = nom de la commande (c-à-d du fichier)
 %1 = premier paramètre
 %n = nième paramètre

Exemple :

Soit le fichier « Affich.bat » contenant les instructions suivantes :

@echo off

Echo exécution du fichier %0

Echo vous avez choisi les couleurs : %1 et %2



Exécution du fichier Affich.bat en fournissant deux paramètres : **C:\Affich rouge noir**

%0 = Affich : nom du fichier de commande

%1 = rouge : premier paramètre

%2 = noir : deuxième paramètre

- **Créer un batch nommé param qui affiche « Hello World » puis affiche « Bienvenue NomPrénom » avec Nom et Prénom deux paramètres.**

2) La programmation batch

- **Le code de sortie d'une commande :**

A chaque saisie d'une commande, cette dernière produit un code de sortie qui peut être :

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.

Exemple :

Dir abc

Echo %errorlevel%

Si le fichier abc existe alors le contenu de errorlevel sera 0, sinon la commande dir va renvoyer un message d'erreur et le contenu de errorlevel sera 1.

- Certaines commandes supplémentaires aux commandes DOS, ont été conçues spécifiquement à la programmation batch : **Rem, Goto, If, Shift, For, Call, Pause, Choice**

❖ La commande REM :

- Cette commande permet d'ajouter un *commentaire* dans un fichier.
- Echo pour la notification utilisateur et Rem pour la documentation batch.

Exemple : Rem ceci est un commentaire

❖ La commande GOTO :

- L'instruction *goto* est une instruction de branchement sur une *étiquette (label)*.

Syntaxe : **GOTO Label**

.....

: Label

.....

- Le label est une chaîne de 1 à 8 caractères alphanumériques. Il apparaît en position d'argument dans l'instruction GOTO.
- Il est utilisé comme étiquette sur une ligne de commande, il doit être précédé de deux points « : »

Exemple : **GOTO FIN**

.....

:FIN

.....

- Si le label n'est pas défini en position d'étiquette lors d'un GOTO, l'exécution du fichier est stoppée. Le message « label not found » s'affiche.

- **En utilisant les étiquettes, dans un fichier parcour.bat créer une boucle qui permet de parcourir les valeurs de 1 jusqu'à 10 puis quitte la boucle.**

❖ La commande IF

- L'instruction **IF** permet l'exécution conditionnelle des commandes MS-DOS.
- Un **if** ne peut être suivi que d'une seule commande (ou plusieurs concaténés par **&**)



Syntaxe : **IF [NOT] Condition Commande1 ELSE Commande2**

- **NOT** : spécifie que l'exécution de la commande est faite si la condition est fausse.
- Tester l'égalité par "=="

Exemple : IF %1 == STOP GOTO FIN

- Tester l'existence d'un fichier ou pas : « **EXIST** » ou « **NOTEXIST** »

Exemple: IF NOT EXIST c:\langage\pascal\%1.pas GOTO ERROR

- Tester l'existence d'un paramètre : **if "%n" == ""**
- A partir de WNT, il est possible d'utiliser **ELSE** avec IF.
- IF et ELSE doivent être sur la même ligne sinon on utilise les ()

Syntaxe : **If condition1 (**
 Commande1
) else if condition2 (
 Commande2
) else(
 Commande3
)

Exemple : *set /p choix= "Que voulez vous faire ? (inscrivez le numéro 1,2 ou 3)" :*
 if "%choix%"=="1" (
 echo bonjour
) else if "%choix%"=="2" (
 echocoucou
) else if "%choix%"=="3" (
 echoaurevoir
) else echo vous n'avez pas inscrit le bon numero

- Les opérateurs possibles :

chaîne1 == chaîne 2	Si chaîne1 est identique à chaîne2
chaîne1 EQU chaîne 2	Si chaîne1 est identique à chaîne2
chaîne1 NEQ chaîne 2	Si chaîne1 est différente de chaîne2
chaîne1 LSS chaîne 2	Si chaîne1 est plus petite que chaîne2
chaîne1 LEQ chaîne 2	Si chaîne1 est inférieur ou égale à chaîne2
chaîne1 GTR chaîne 2	Si chaîne1 est supérieur à chaîne2
chaîne1 GEQ chaîne 2	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.*
- **Créer un fichier de commande affichant un message pour dire si le fichier passé en paramètre est présent ou non dans le répertoire courant.** (Exemple d'appel : **trouve toto.txt**)
- **Ecrire un batch qui permet de calculer et afficher la somme de deux paramètres.**

❖ **La commande SHIFT**

L'instruction **SHIFT** permet de faire un décalage de paramètres.

Elle permet :

- Soit d'utiliser plus de 9 paramètres par simple décalage
- Soit de parcourir les paramètres passés
- Soit de traiter les paramètres sans en connaître le nombre



Après chaque instruction SHIFT, le paramètre réel 1 disparaît, le paramètre réel 2 devient 1 (.... et ainsi de suite pour le reste des paramètres)

- **Créer un batch qui affiche la somme de tous les arguments de la commande.**

❖ La commande FOR

- L'instruction FOR exécute une commande sur une chaque valeur d'un ensemble.
Syntaxe : **FOR %variable IN (Ensemble) DO Commande [Paramètres]**
 - ✓ **Ensemble** : ensemble de valeurs, de fichiers séparés par des espaces (caractères génériques permis) qui sont parcourus par la variable
 - ✓ **Commande** : commande à exécuter pour chaque commande
 - ✓ **Paramètres** : liste des paramètres ou options pour la commande spécifiée.
- Dans un fichier de commandes, il faut spécifier %%**variable** au lieu de %variable pour faire la différence avec les paramètres du fichier.

Exemple : Rem ce programme affiche bêtement de 1 à 5

```
For %%i in (1 2 3 4 5) do echo %%i
```

Rem ce programme affiche le contenu des fichiers batch de deux répertoires

```
For %%i in (c:\rep1\ c:\rep2\) do type %%i\*.bat
```

- **Après avoir exécuté ces exemples par un fichier batch, exécutez-les directement sur console.**

❖ La commande CALL

- La commande **CALL** est une instruction qui permet :
 - d'appeler un script batch dans le script principal. (*Call chemin arguments*)
 - ou une sous-routine située à un label particulier du fichier. (*Call :label arguments*)
- Après la commande, l'exécution se poursuit à l'instruction suivante.

Exemple : *CALL :affiche C:\Windows*

```
CALL :affiche C:\aloulou
```

```
GOTO :eof
```

```
:affiche
```

```
ECHO Le contenu du répertoire %1 est :
```

```
DIR %1
```

```
GOTO :eof
```

- Pour sortir du batch ou terminer la sous-routine, le label spécial **:eof** est utilisé.

❖ La commande PAUSE

- L'instruction **PAUSE** suspend l'exécution du fichier de commande et affiche "*Appuyer sur une touche pour continuer...*"
- **PAUSE** sert pour donner à l'opérateur :
 - le temps de réfléchir
 - la possibilité de changer une disquette
 - la possibilité d'interrompre l'exécution par CTRL-C



❖ La commande CHOICE

- La commande CHOICE permet à l'utilisateur de choisir un objet parmi une liste de choix et retourne l'indice du choix sélectionné.
 - Cette commande possède quelques paramètres intéressants, tels que :
 - **/C choix** : spécifie la liste des choix (par *défaut c'est Y/N*)
Exemple : CHOICE /C ABC
 - **/T temps** : c'est le nombre de secondes d'attentes avant de faire un choix par défaut.
Exemple : CHOICE/T 10 → provoque une attente de 10 secondes
CHOICE /T 0 → ne provoque pas d'attente
 - **/D choix** : spécifie le choix par défaut après un temps d'attente en secondes. Le choix doit être parmi les choix de /C et doit être utilisé avec /T.
Exemple : CHOICE /C ABC /D A /T 10
 - **/M message** : affiche un message avant le prompt.
Exemple : CHOICE /M "Faites un choix" → ici les guillemets sont obligatoires
- **Ecrire un fichier batch nommé `supprime.bat` qui permet à l'utilisateur de choisir entre la suppression de 3 fichiers : `aide.txt`, `index.txt` et `visite.txt`. Si l'utilisateur ne fait aucun choix après un délai de 5 secondes alors c'est le fichier `aide.txt` qui sera supprimé.**

3) Exercices**Exercice 1 :**

Créer un fichier `chiffre.bat` qui prend un chiffre comme argument et l'affiche en toutes lettres.

Exercice 2 :

Faire le même exercice précédent avec plusieurs arguments.

Exercice 3 :

Créer un batch qui affiche le plus grand de tous les arguments

Exercice 4 :

Ecrire un script batch qui permet de copier un ensemble de fichiers `.txt` de plusieurs sources différentes vers un seul emplacement `D:\Mestextes`.

