TP1: Introduction au Shell Windows

Consignes

L'objectif de ce TP est de vous familiariser avec le Shell Windows en étudiants les commandes de base. Compléter le document en répondant aux questions, ce qui vous aidera à mémoriser les commandes.
Outils utilisés : Éditeur de texte, Notepad++ recommandé.

1 Utilisation du shell

Lancer l'interpréteur de commande : touche "windows" + R, et taper "cmd" puis "Entrée".

Q1.1. Affichez la date : **`DATE`**. Donner la commande permettant d'avoir la date sans invitation à en entrer une nouvelle :

Q1.2. Avec la commande **`VER`**, donner la version de l'OS installé :

Q1.3. Affichez la liste des commandes DOS disponibles, à l'aide de la commande **`HELP`**

Q1.4. Quelles informations donne la commande **`HELP DIR`** :

Q1.5. Y a-t-il une différence avec **`DIR /?`** :

Q1.6. Affichez l'aide de la commande « prompt » : **`HELP PROMPT`**.
Quelles informations vous sont fournies par la commande **`SET PROMPT`** ?
Notez les informations affichées, et déterminez leur signification :

Q1.7. Modifiez le prompt par la commande **`PROMPT $P$_$T$S`**

Q1.8. Modifiez le prompt pour obtenir l'affichage **`IUTRT>`** et donner la commande correspondante :

Q1.9. Quitter le shell et relancez le. Les modifications du prompt sont elles persistantes après fermeture de la console ?

Q1.10. Essayez et comparez les commandes
`echo bonjour` et **`echo bonjour > bonjour.txt`**
Que fait la deuxième version :

2 Recherche de fichiers

Q2.1. Donner la commande qui me donne le dossier courant :

Q2.2. Basculer sur **`C:`** et positionnez vous dans le répertoire
`\Windows\system32`
En exploitant la commande **`DIR`**, affichez la liste des fichiers ayant la caractéristique suivante. Donnez la commande et le nombre de fichiers trouvés :

Q2.1. Fichiers dont le nom commence par **`TS`**, et ayant l'extension **`DLL`** :

Q2.2. Fichiers .DLL dont le nom fait 7 caractères dont la 4e lettre est un 'q' :

Q2.3. Affichage de la liste des fichiers classés par ordre alphabétique sur le nom :

Q2.4. Affichage de la liste des fichiers classés par ordre alphabétique sur l'extension :

Q2.5. Affichage de la liste des fichiers classés par ordre chronologique :

Q2.3. Sans changer le dossier actif, donner la commande pour créer un dossier **`T:\M1105`** :

Q2.4. Enregistrez dans le fichier **T:\M1105\catalogue.txt** la liste des fichiers du répertoire **C:\Windows**, classés par ordre alphabétique sur le nom (utiliser une redirection) :

Q2.5. Saisissez la commande **T:\M1105\catalogue.txt** (éventuellement en rappelant la dernière commande et en l'éditant). Quelle application s'ouvre ? :

Q2.6. Qu'en déduisez vous ?

3 Copie et déplacement de fichiers et dossiers

Le Shell Windows dispose de deux commandes de copie natives, **COPY** et **XCOPY**. La première permet de copier des fichiers multiples (via des caractères génériques), la 2e étant dédiée à de la copie de dossiers entiers.

Q3.1. Positionnez le répertoire actif sur **T:\M1105\test1**. Donner la commande permettant de créer depuis cet endroit un dossier **T:\M1105\test2**, en utilisant un chemin relatif :

Q3.2. Créer deux fichiers contenant une simple ligne de texte avec les commandes suivantes :

```
echo aaa > aaa.txt
echo bbb > bbb.txt
```

Donner la commande pour copier ces deux fichiers dans le dossier **T:\M1105\test2**, en utilisant un chemin relatif, la commande COPY et des caractères génériques :

Q3.3. La commande précédente affiche **2 fichier(s) copié(s).** En utilisant une redirection, donner la commande de copie en faisant en sorte que ce message ne s'affiche pas et qu'il n'y ait pas de demande de confirmation d'écrasement :

Q3.4. Concatener ces deux fichiers avec la commande
copy aaa.txt+bbb.txt T:\M1105\concat1.txt
Donner la taille en octet des trois fichiers :
aaa.txt :
bbb.txt :
concat1.txt :

Q3.5. Pour les deux premiers, d'où viennent les octets "en trop" ? (ouvrez les dans un éditeur)

Q3.6. Pour copier des dossiers entiers, il faut utiliser la commande **xcopy**, qui a de nombreuses options. A partir de l'aide de cette commande, donner la commande pour copier l'intégralité du dossier **T:\M1105** dans un dossier **T:\M1105_copy** :

Q3.7. Créer un nouveau fichier **ccc.txt** dans le dossier **T:\M1105\test1** et donner la commande permettant de faire la même copie que précédemment mais cette fois de façon **incrémentale**, c'est à dire en ne copiant que les fichiers qui ont été modifiés ou ajoutés :

Q3.8. Donner la commande permettant d'effacer ce fichier **ccc.txt**, quel que soit le dossier actif :

4 Attributs de fichiers

Deux types d'attributs de fichiers sont disponibles avec Windows :

— Les attributs "legacy" hérités du système de fichier "FAT" : chaque fichier n'a que quatre attributs : R (*Read Only*), A (*Archive*), S (*System*), H (*Hidden*).

— Les permissions évolués du système de fichiers NTFS : ACL (*Access Control List*).

Les premiers sont éditables avec la commande **attrib**, les seconds nécessitent un programme externe **xcacls**, souvent fourni de façon séparée ("*Windows resource kit*").

Q4.1. Avec la commande **attrib**, donner la commande permettant de rendre caché le fichier **T:\M1105\test1\ccc.txt** :

Ce fichier apparaît-il avec la commande **dir** ? :

Apparaît-il dans l'explorateur de fichier ? :

Sous quelle condition ? :

Q4.2. Donner la commande permettant d'enlever cet attribut "caché" :

5 Écriture de scripts

Q5.1. Créer un fichier texte nommé **T:\M1105\TP1_a.bat** et y taper les lignes suivante :

```
echo Bonjour
pause
cls
echo vous etes %USERNAME%
```

Q5.2. Lancer ce script depuis le dossier **T:\M1105**, en tapant son nom (inutile de préciser l'extension). Que constatez vous ? :

Q5.3. Pour éviter ce double affichage, deux solutions sont possibles

— Préfixer chaque commande avec le symbole **@** :
@echo "Bonjour"

— Désactiver l'écho des commande en placant au début du fichier la ligne **@echo OFF**.

L'interêt de cette deuxième méthode est que en cas de dysfonctionnement (phase de deboggage), on pourra réactiver globalement l'écho de chaque commande en remplaçant temporairement cette ligne par : **@echo ON**.

Q5.4. En utilisant ce que vous avez vu, proposer une solution pour remplacer le message de la commande **PAUSE** par un affichage personnalisé (indice : utiliser une redirection ver le néant...)

Q5.5. Se placer dans le dossier **T:** et taper **TP1_a**. Le script s'exécute-t-il ? :

Q5.6. Une variable d'environnement spéciale nommée **PATH** contient la liste des chemins où l'OS doit aller chercher les commandes et programmes. On peut l'afficher avec la commande du même nom, ou en tapant **set P**. Afficher cette variable. Y ajouter le chemin **T:\M1105** avec la ligne : **PATH=%PATH%;T:\M1105;**. Rappeler la commande précédente (fleche curseur) et la relancer. Le script s'exécute-t-il ? :

Q5.7. Taper la commande **exit** puis relancer l'interpréteur. La variable PATH a-t-elle été conservée ? :

6 Structuration des scripts

Windows permet une structuration minimale d'un script de deux façons :

— Via des "sauts absolus" avec la commande **GOTO** qui déroute l'exécution à un autre endroit du script.

— Via des pseudos-fonctions avec retour automatique avec la commande **CALL** qui déroute l'exécution à un autre endroit du script, puis permet de "revenir" juste après la ligne de l'appel. Le retour se fera avec la commande **goto :eof** (ce dernier symbole est donc un label "réservé").

Avec ces deux commandes, on spécifie la cible avec un **label**, qui doit exister dans le fichier. Par exemple :

```
echo "depart vers piano"
goto :piano
echo "cette ligne n'est jamais executee"

:piano
echo "piano: depart vers guitare"
goto :guitare
echo "jamais executee non plus"

:guitare
:: fin
```

Q6.1. Saisir le programme suivant dans un fichier **T:\M1105\tp1_3.bat** puis l'exécuter.

```
call :piano
call :piano
call :piano
echo "compt= %compt%"
goto :eof :: fin du script

:piano
echo "piano: appel %compt%"
set /a compt= %compt%+1
goto :eof
```

Quelle valeur s'affiche à la fin :

Q6.2. Relancer le programme une 2e fois. Que constatez vous ? :

Q6.3. Coment faire pour la réinitialiser ?

7 Arguments de scripts

Comme tout programme en ligne de commande, on peut passer des arguments aux scripts.

Q7.1. Créer un fichier en tapant :

```
@echo @echo tp1_2: arg1=%1 arg2=%2 > T:\M1105\tp1_2.bat
```

Puis l'exécuter avec la ligne

```
tp1_2.bat bonjour monsieur
```

et vérifier le résultat.

Cette technique peut aussi être utilisée pour passer des arguments à des sous-programmes. La ligne **CALL :mon_sp1 aaa bbb**
va dérouter l'exécution à l'étiquette **:mon_sp1** et on pourra récuperer les deux arguments dans les paramètres positionnels **%1** et **%2**.

8 Exercice de synthèse

Exercice : dans une entreprise, deux commerciaux (Pierre et Paul) doivent travailler sur un disque partagé **T:**, avoir chacun leurs dossier, tout en ayant accès aux dossiers de l'autre.

Écrire un script **T:\M1105\tp1_creation.bat** qui crée dans le dossier **T:\users** un dossier pour chacun à leur nom, et dans chacun des deux l'arborescence suivante :

```
<nom>
├── commandes
│   ├── en_cours
│   └── soldées
├── courrier
└── factures
    ├── en_cours
    └── soldées
```

Dans chaque répertoire, vous devez insérer un fichier nommé **log.txt** contenant la date et l'heure de création.

Vous utilisez un sous-programme de création de l'arborescence, auquel vous passerez en argument le nom de l'utilisateur :

```
call :sp1 Pierre
call :sp1 Paul
```

TP1 : exercices supplémentaires

Q8.1. Ecrire un script qui prend en argument une chaîne de caractères et qui affiche OUI si cette chaîne correspond au nom de la machine et NON dans le cas contraire, en affichant ce nom.

Q8.2. Ecrire un script qui va créer une archive **`T:\M1103`** de tout le contenu de votre dossier **`T:\M1103`**. Vous utiliserez le programme 7-zip, installé sur la machine.

Boucle "For" : L'interpréteur Windows est doté d'une commande "for" qui peut prendre plusieurs formes. La version la plus simple permet d'énumérer des fichiers :
`FOR %variable IN (ensemble) DO commande`

Q8.3. Ecrire un programme qui va compter le nombre de fichiers ayant l'extension donnée en argument dans le dossier **`c:\Windows`**, et qui afficher uniquement le résultat final. Le script devra vérifier quel'argument est non vide et appartient à l'une de celle-ci :DLL, INI ou TXT. Il devra afficher une erreur dans les autres cas.

Références

— Microsoft : `https://technet.microsoft.com/en-us/library/cc754340.aspx`
— Microsoft (version française) : `https://technet.microsoft.com/fr-fr/library/cc754340(v=ws.10).aspx`
— Simon Sheppard : `http://ss64.com/nt/`