

Travail pratique 3

Programmation console Powershell

Note sur 15 points (15% de la note finale)

Date limite : **19 décembre 2018**

Dans ce travail, vous serez amené à créer des scripts Powershell pour Windows. Aussi, vous apprendrez à utiliser Vagrant, Docker et des serveurs Web.

Vous devez être en équipe de 2 (si ce n'est pas possible, consultez le professeur). Les critères d'évaluation détaillés de ce travail seront disponibles sur LEA.

A) Outil de recherche dans l'observateur d'événement (2 points)

Faites un programme Powershell appelé `search_eventlog.ps1` qui permet de rechercher dans l'observateur d'événement et d'agir dessus.

Si vous passez un mot-clé en première paramètre, vous obtenez l'ensemble des entrées d'événement contenant le mot-clé en paramètre. Par exemple :

```
> & ./search_eventlog.ps1 update
```

```
22 Dec 18 22:42 Information Microsoft-Windows... 4097 Successful auto  
update of third-party root cert...
```

```
21 Dec 18 22:42 Information Microsoft-Windows... 4097 Successful auto  
update of third-party root cert...
```

```
[...]
```

Si vous ajoutez un deuxième paramètre, tous les messages seront enregistrés dans le fichier en deuxième paramètre. Voici un exemple :

```
> & ./search_eventlog.ps1 mot-cle fichier.txt
```

```
Enregistré dans le fichier "fichier.txt"
```

Si vous ne spécifiez aucun paramètre, un message d'erreur s'affiche pour informer qu'il faut au moins un paramètre.

```
> & ./search_eventlog.ps1
```

Vous devez spécifier au moins le mot-clé de la recherche.

Pour ce script, **il n'y a aucun menu à afficher**. Les fonctionnalités du script dépendent des paramètres fournis à l'exécution de la commande.

Servez-vous des commandes

Get-EventLog

et

Get-ChildItem -Recurse -Include

Vous devez vous assurer que la recherche se fasse sur toutes les catégories d'événements (« LogName ») disponibles.

B) Rechercher et remplacer (3 points)

Faites un script Powershell qui permet de rechercher et remplacer une chaîne de caractère par une autre dans tous les fichiers textes courants et tous les fichiers des sous-dossiers récursivement¹.

Le script s'utilise avec 2 paramètres : la chaîne de caractères à rechercher et la chaîne de caractères. La recherche se fait seulement dans le dossier de l'exécution du script.

Avant d'effectuer un remplacement, le script doit poser la question à l'utilisateur s'il accepte le remplacement. Voir des exemples d'utilisation :

```
> & ./search_replace.ps1 "Monsieur" "Madame"
```

Remplacer la chaîne "Monsieur" dans le fichier
.\Entreprise\Employe1\document1.txt ? (o/n) o

Le remplacement est effectué.

Remplacer la chaîne "Monsieur" dans le fichier
.\Entreprise\Employe2\document2.csv ? (o/n) n

Aucune changement n'est effectué.

Il y a eu 2 occurrence(s) trouvée(s) et 1 remplacement(s).

```
> & ./search_replace.ps1 "Monsieur"
```

Vous devez spécifier la chaîne de remplacement pour l'exécution.

```
> & ./search_replace.ps1
```

Vous devez spécifier la chaîne recherchée et la chaîne de remplacement pour l'exécution.

Pour ce script, **il n'y a aucun menu à afficher**. Les fonctionnalités du script dépendent des paramètres fournis à l'exécution de la commande.

¹ Ce n'est pas seulement pour les fichiers TXT. La recherche doit se faire sur tous les fichiers quel que soit son format.

C) Recherche et Installateur MSI (6 points)

Faites un script Powershell qui est similaire au script BAT réalisé au travail pratique 2 et qui permet l'automatisation d'installation de logiciels en format MSI. Ce script s'appelle **installSoft.ps1**. A la différence du script du travail pratique 2, la liste des fichiers MSI et Chocolatey sont stockés dans la base de registre de l'ordinateur (servez-vous du programme **regedit** pour vérifier). Voici des cas d'utilisation (dépendant des paramètres):

1. > & ./installSoft.ps1 /add_to_registry_key git.install

Cette commande enregistre le chemin d'accès au fichier `mon_installateur.msi` dans la base de registre. En fait, plutôt que d'enregistrer la liste des fichiers MSI ou Choco dans un fichier texte comme pour le TP 2, on se sert de la base de registre cette fois-ci. Évidemment, il pourrait y avoir plusieurs enregistrements. Il faut donc que le script enregistre plusieurs chemins d'accès dans la même clé de registre. La clé de registre doit être enregistré à cet endroit :

`"HKEY_CURRENT_USER\Software\InstallSoft"`

La liste des fichiers MSI et Chocolatey sont enregistrés dans la même clé en format « string » et les fichiers sont séparés par un point-virgule (;) comme par exemple :

`"7zip.install;atom;monapplication.msi;git.install;openssh;puppet-agent;putty.msi"`

2. > & ./installSoft.ps1 /install

Cette commande installe tous les fichiers MSI et Chocolatey qui se trouvent dans la base de registre. Quand l'installation est terminée, la clé de registre reste inchangée.

3. > & ./installSoft.ps1 /show_registry_key

Content : `"7zip.install;atom;monapplication.msi;git.install;openssh;puppet-agent;putty.msi"`

Cette commande affiche le contenu de la clé de registre contenant la liste des chemins d'accès aux fichiers MSI et Chocolatey.

4. > & ./installSoft.ps1 /remove_all_registry_key

Cette commande supprime la clé de registre contenant la liste de tous les chemins d'accès aux fichiers MSI ou Chocolatey.

5. > & ./InstallSoft.ps1 /remove_from_registry_key mon_installateur.msi

Cette commande enlève le fichier « `mon_installateur.msi` » de la clé de registre de la liste des fichiers MSI ou Chocolatey à gérer.

6. > & ./InstallSoft.ps1 /uninstall

va exécuter la commande `msiexec` ou `choco` en mode silencieux pour désinstaller les logiciels (noms des fichiers MSI) disponibles dans la clé de la base de registre.

7. > & ./InstallSoft.ps1 /update

va exécuter la commande `msiexec` ou `choco` en mode silencieux pour mettre à jour les logiciels (noms des fichiers MSI) disponibles dans la clé de la base de registre.

8. > & ./InstallSoft.ps1 /help

Cette commande affiche un texte d'aide pour la commande `InstallSoft`. Tous les paramètres ci-haut doivent apparaître dans le texte d'aide.

9. > & ./InstallSoft.ps1

Sans aucun paramètre, la commande affiche le texte d'aide comme avec le paramètre `/help`.

Pour ce script, **il n'y a aucun menu à afficher**. Les fonctionnalités du script dépendent des paramètres fournis à l'exécution de la commande.

Les fichiers MSI ne sont pas à remettre. Seuls les fichiers Powershell sont à remettre (compressés en format ZIP) sur LEA.

D) Vérification de l'installation d'un serveur Web (4 points)

Faites un script Powershell appelé `scan_webserver.ps1` qui

1. vérifie si un serveur Web est bien actif,
2. détermine le type de système d'exploitation
3. et détermine le logiciel du serveur Web (IIS, Apache, Nginx).

Pour réaliser vos expérimentations, installez le serveur Web NGinx expliqué dans l'exercice 17, un serveur Apache expliqué dans l'exercice 21 et un serveur IIS expliqué dans l'exercice 22.

Voici les étapes :

1. Installez Virtualbox, Vagrant et le serveur [Scotch-io](#) comme expliqué dans l'exercice 17. Le serveur devra être accessible à <http://192.168.33.10>.
2. À l'aide de Chocolatey, votre script doit installer les applications [curl](#) et [nmap](#) seulement si ces applications ne sont pas déjà installées.
3. Installez Docker comme expliqué dans l'exercice 21. Le serveur Apache doit être accessible à l'adresse fournie par la commande `docker-machine` qui pourrait être par exemple <http://192.168.99.102>
4. Installez le serveur IIS via Vagrant comme expliqué dans l'exercice 22 disponible [ici](#). Le serveur IIS doit être accessible à l'adresse <http://127.0.0.1:1025>

Il n'est pas nécessaire d'exécuter les 3 serveurs Web en même temps. Un après l'autre suffit.

Votre script vérifie si le serveur est actif en exécutant la commande :

```
> curl -I -L http://192.168.33.10:80
```

ou

```
> curl -I -L http://192.168.99.102
```

(l'adresse peut être différente)

ou

```
> curl -I -L http://127.0.0.1:1025
```

Votre script vérifie le système d'exploitation où le serveur Web est installé en exécutant la commande :

```
> nmap -O 127.0.0.1
```

ou

```
> nmap -O 192.168.33.10
```

Votre script vérifie le nom du serveur Web installé en exécutant la commande :

```
> nmap -sV -P0 -p1025 127.0.0.1
```

ou

```
> nmap -sV -P0 -p80 192.168.99.102
```

 (l'adresse peut être différente)

ou

```
> nmap -sV -P0 -p80 192.168.33.10
```

Votre script doit s'exécuter comme par exemple :

```
> & ./scan_webserver.ps1 127.0.0.1 1025
```

Serveur actif: oui

Système d'exploitation: Windows

Application Web: IIS

ou

```
> & ./scan_webserver.ps1 192.168.99.102 80
```

 (l'adresse peut être différente)

Serveur actif: oui

Système d'exploitation: Linux

Application Web: Apache

ou

```
> & ./scan_webserver.ps1 192.168.33.10 80
```

Serveur actif: oui

Système d'exploitation: Linux

Application Web: Nginx

```
> & ./scan_webserver.ps1
```

Il manque les paramètres de l'adresse IP et le port pour scanner.

Pour ce script, **il n'y a aucun menu à afficher**. Ce script s'exécute dans une console Powershell avec les 2 paramètres obligatoires.

Pour que les serveurs Web soient accessibles et fonctionnent bien, il faut arrêter tout parefeu et antivirus de votre système d'exploitation.