Scripting en Bash et Powershell

J.L. DAMOISEAUX
IUT Aix-Marseille, Aix-Marseille Université
163 av de Luminy
case 920, 13288 Marseille Cedex 9
+33 491 177 927
jean-luc.damoiseaux@univ-amu.fr

D. MANOUKIAN
IUT Aix-Marseille, Aix-Marseille Université
163 av de Luminy
case 920, 13288 Marseille Cedex 9
+33 491 177 927
damien.manoukian@univ-amu.fr

RÉSUMÉ
Dans ce papier nous présentons notre approche pédagogique pour l'enseignement du module M3206 "Automatisation des tâches d'administration". Après une brève introduction de la problématique, nous expliquerons notre approche pédagogique, puis nos choix algorithmiques au travers d'exemples de travaux pratiques. Nous conclurons en donnant le ressenti des étudiants.

CCS Concepts
• Applied computing → Education → Computer-managed instruction

https://dl.acm.org/ccs/ccs.cfm.

Mots clefs
Scripting ; Bash ; Powershell ;

1. INTRODUCTION
En charge du module M3206 "Automatisation des tâches d'administration", nous sommes rapidement arrivés au constat suivant :

- la grande majorité de nos étudiants préférait les "réseaux" (et plus précisément la configuration d'équipements) à la programmation ;
- de part le faible volume d'heures officiellement disponibles (30H), et du fait que les étudiants devaient être en mesure d'automatiser des tâches sous Linux et Windows, nous ne pouvions pas entrer dans des concepts avancés de la programmation.

Aussi, afin de motiver nos étudiants, tout en satisfaisant au PPN et son financement, tout en répondant aux contraintes de mise en œuvre, nous avons donc réalisé un ensemble de travaux pratiques :

- le plus souvent inspirés de problématiques "réseau" rencontrées au cours des suivis de stages ;
- basé sur un algorithme simple et transposable facilement dans les deux langages.

Dans cet article, nous détaillerons donc notre approche pédagogique au travers de l'articulation de nos travaux pratiques, de l'algorithme retenu, des sujets proposés et de leurs développements possibles.

2. APPROCHE PÉDAGOGIQUE
2.1 Déroulé pédagogique
Le choix de la présentation du Bash [1] et du Powershell [2], couplé à notre volonté d'accentuer la pratique du langage, nous ont conduit à un découpage de notre module différent de celui proposé dans le PPN. Ainsi, nous avons moins d'heures de cours et aucune séance de travail dirigé qui, compte tenu de la taille de nos salles informatiques, se traduiraient obligatoirement par de la programmation sur papier. Notre découpage pédagogique, identique pour le Bash et le Powershell, est donc constitué d'une heure de cours, dix heures de travaux pratiques (à raison de 2H par semaine) et d'une heure d'examen écrit (note qui viendra compléter celles mises au cours des travaux pratiques).

Table 1. Comparaison des déroulés pédagogiques du module M3206

	PPN	Bash	Powershell
Cours	6H	1H	1H
TD	6H		
TP	18H	10H	10H
Examen		1H	1H
Total	**30H**	**12H**	**12H**

2.2 Contenu du cours
Pour le Bash et le Powershell, nous limitons notre étude du langage aux structures de données simples (variables et tableau unidimensionnel), aux opérateurs arithmétiques et de comparaison, et aux structures de contrôle de base (le si-alors-sinon, l'aiguillage, et les différentes boucles).

Des notions importantes comme la portée d'une variable, les fonctions, ne sont absolument pas abordés. Mais, pour chacun des deux langages, quelques spécificités, nécessaires aux étudiants en travaux pratiques, sont traitées.
Pour le Bash, un rappel est fait sur l'effet des différents caractères spéciaux, les redirections et l'enchaînement de processus. Des explications sont données pour le traitement des arguments de la ligne de commande.
Pour le Powershell DEMANDER à DAMIEN

2.3 Liste des travaux pratiques
Le cœur de nos travaux pratiques est composé de deux sujets. Le premier, où les étudiants vont s'approprier les éléments de base du langage, est composé de la suite des petits exercices suivants :

- détermination de la classe d'une adresse IP saisie au clavier ou donnée en argument de la ligne de commande ;
- découverte des machines présentes sur un réseau (Figure 1.) ;
- interrogation du DNS pour connaitre les adresses/noms symbolique d'une liste de sites/adresses contenue dans un fichier ;

 # script2.ps1

```
$ping = new-object System.Net.NetworkInformation.Ping
for ($i = 1; $i -lt 255; $i++) {
    $iphost = "192.168.1.$i"
    $onlinetest = Test-Connection -computername $iphost -Count
1 -quiet
    switch ($onlinetest) {
        $true { $available = "Disponible" }
        Default { $available = "Indisponible" }
    }
    Write-Host "La machine $iphost est $available"
    }
```

```
#!/bin/bash
for ((i=1; i<=254; i=i+1)) do
ping -c1 192.168.1.$i > /dev/null
if (($? == 0)); then
    echo "la machine 192.168.1.$i est sur le reseau"
fi
done
```

Figure 2. Balayage d'un réseau de classe C en Powershell et Bash

Le second sujet est beaucoup plus conséquent puisqu'il s'agit de créer un ensemble de comptes utilisateurs sur chacun des systèmes d'exploitation en question. Des prolongations à ce sujet sont possibles et permettent de faire le lien avec les modules M1106, M2105 et M2106 du PPN.

Dans le cadre du Bash, les étudiants, installeront un serveur apache et écriront une page html appelant un script cgi permettant la réinitialisation de leur mot de passe. Une fois ceci fait, pour les sensibiliser à la sécurité et à l'importance de maintenir à jour un système, ils mettront en œuvre l'attaque Shellshock [3] au travers d'une requête http dont la chaine `User-Agent` aura quelque peu été modifiée.

Dans le cadre du powershell, les étudiants pourront compléter le script en supprimant sur demande un compte, ou en faisant en sorte que l'utilisateur soit obligé de changer son mot de passe à la prochaine connexion.

Enfin, pour les étudiants plus à l'aise avec la programmation, nous proposons d'autres sujets comme l'exploitation d'un fichier log d'un firewall, la supervision d'un réseau via le protocole snmp, la sauvegarde automatisée d'une répertoire/base de données, etc. Là encore, il est souvent possible de faire le lien avec d'autres modules du PPN, par exemple en demandant aux étudiants de générer une sortie des résultats au format html.

3. PRINCIPES GÉNÉRAUX DE NOS SUJETS

A l'exception de la première série d'exercices, tous les autres sujets utiliseront comme structure de donnée un tableau unidimensionnel, et s'appuieront sensiblement sur un même algorithme constitué d'une boucle et de tests. Même s'il existe certainement des solutions plus élégantes ou performantes à nos sujets, nous voulons donner un cadre fixe à nos étudiants afin qu'ils puissent se l'approprier et le réutiliser à chaque nouvel exercice.

L'idée directrice de ces travaux pratiques est donc de remplir un tableau à partir d'éléments issus d'un fichier ou du résultat d'une commande, puis de parcourir un à un ces éléments pour les tester et agir en conséquence.

Une fois cela fait, les étudiants enrichiront le programme par l'ajout de fonctionnalités et/ou le contrôle des paramètres d'entrée (Figure 2.).

```
⑤ Si les paramètres d'entrée sont corrects alors
    ① Remplir le tableau à partir du fichier
    ② Pour chacun des éléments du tableau faire
        ③ Si une condition est vraie alors
            réaliser le traitement associé
        sinon
            réaliser un autre traitement
    ④ Afficher et/ou sauvegarder les résultats
```

Figure 2. Algorithme de base et ordre dans lequel les étudiants le mettront en œuvre

Nous allons maintenant illustrer les principaux points de cet algorithme au travers d'exemples, et en faisant un focus sur les points importants assurant une meilleure réussite de l'étudiant.

3.1 Remplissage du tableau de données

Ce premier point est illustré par le sujet sur la création automatique de compte utilisateurs. Les données sources sont stockées dans un fichier csv où chaque ligne est de la forme

nom;groupe;identifiant

Dans le cadre de cet exercice nous demandons aux étudiants de répartir les données de ce fichier en trois tableaux *noms*, *groupes* et *identifiants*. Pour simplifier les traitements futurs et donc minimiser les erreurs des étudiants, il est important que chaque ligne de notre fichier soit complète, ce qui aura pour conséquence l'utilisation d'un seul indice pour parcourir nos trois tableaux.

En Powershell, le remplissage du tableau *noms* se fera par les commandes :

```
$csvFile = "users.csv"
$noms = Import-Csv $csvFile -Delimiter ';'
$noms | Add-Member -MemberType NoteProperty -Name
_add -Value 0
$noms | Add-Member -MemberType NoteProperty -Name
_pwd -Value ""
```

où *A COMPLETER PAR DAMIEN.*

En Bash, le remplissage du tableau utilisateur se fera par la commande :

```
noms=(`cat ${1} | cut -d';' -f1`)
```

où *${1}* est le nom du fichier csv.

3.2 Traitement du tableau de données

Les étudiants ayant maintenant rempli le tableau de données, ils vont pouvoir, par raffinement successifs, obtenir un programme abouti.

Dans l'exemple de la création de comptes, l'algorithme général de départ pour les phases ② et ③ est (Figure 3.) :

```
② Pour chacun des éléments du tableau noms faire
    ③ Si noms[i] existe alors
            Afficher " noms[i] a déjà un compte"
        sinon
            si groupes[i] n'existe pas alors
                Afficher "il faut créer le groupes[i]"
            Afficher "il faut générer un mot de passe"
            Afficher "il faut créer l'utilisateur nom[i]"
```

Figure 3. Algorithme initial de création de comptes

Nous demandons alors aux étudiants de successivement remplacer tous les affichages par les commandes adéquates (Figure 4.).

```
for ((i=0; i<=${#noms[@]}; i++)) do
    if grep ${noms[i]} /etc/passwd > /dev/null; then
        echo "${noms[i]} existe deja"
    else
        if ! grep ${groupes[i]} /etc/group > /dev/null; then
            groupadd ${groupes[i]}
            mkdir /home/${groupes[i]}
        fi
        mdp=`< /dev/urandom tr -dc '!_A-Z-a-z-0-9' | head -c8`
        useradd -p `mkpasswd $mdp` -m -g ${groupes[i]} -b /home/${groupes[i]} ${noms[i]}
```

Figure 4. Cœur du programme Bash pour la création de comptes

A noter que le fait d'imposer aux étudiants des raffinements successifs de leur programme, ils en arrivent souvent à écrire par eux même un nouveau script pour réinitialiser l'état des comptes et groupes de leur système.

Un deuxième exemple d'application de cet algorithme peut être donné sur le sujet d'analyse d'un fichier de log d'un firewall. Ce fichier se présente sous la forme d'une suite de lignes :

Menace ID IPSrc IPDst

et nous demandons aux étudiants de retrouver toutes les attaques du même type, ou en provenance/à destination d'une machine spécifique. L'utilisation d'un tableau dans lequel on récupère, selon les besoins, toutes les adresses sources ou de destination, fait que le cœur du programme devient très similaire au précédent (Figure 5.).

```
listAdrIP=(`cat menaces.txt | tail -n +2 | sed 's/^[^0-9]*//' | awk '{print $4}'`)
ipAChercher=${2}
i=0
for adrIP in ${listAdrIP[@]}; do
    ((i++))
    if [ $adrIP = $ipAChercher ]; then
        cat menaces.txt | tail -n +2 | tail -n +${i} | head -n 1 | sed 's/[0-9.]//g' | sed 's/[ ]*$//'
    fi
```

Figure 5. Cœur du programme Bash pour l'analyse d'un fichier log

4. CONCLUSION

Cet article est une contribution à l'enseignement du module M3206 du PPN. La structuration proposée, l'orientation donnée aux sujets, leur reprise en Bash et en Powershell, contribuent grandement à la réussite des étudiants dans ce module puisque même les plus faibles en programmation arrivent au bout deux premiers travaux pratiques. Tous les sujets et leur correction sont disponibles sur le site http://

A plus long terme, il serait intéressant de réfléchir à l'utilisation pour ce module du langage Python. En effet, Python est très souvent enseigné dans le secondaire, de plus en plus utilisé dans les entreprises, poussé par un grand constructeur pour la configuration de ces équipements, et multi-plateforme.

5. RÉFÉRENCES

[1] Bash Reference Manual
https://www.gnu.org/software/bash/manual/bash.pdf

[2] PowerShell Documentation
https://docs.microsoft.com/en-us/powershell/.

[3] Shellshock Vulnerability
https://nvd.nist.gov/vuln/detail/CVE-2014-6271