Table des matières

1 Introduction

Les objectifs de la formation

Connaître les caractéristiques des principaux outils de scripting Unix/Linux
Savoir lire des scripts Unix/Linux écrits en Shell, Perl, Python Ruby ou AWK
Être capable d'écrire des scripts simples d'exploitation Unix/Linux
Comprendre comment choisir l'outil le plus adapté pour résoudre un problème particulier

Prez perso et demander le niveau des gens

2 Le Shell - les bases

2.1 Qu'est ce qu'un shell

Un shell Unix est une interface homme machine (IHM) en ligne de commande (CLI). Il fournit à la fois un langage de commandes interactives et un langage de scripting. Le shell traite des commandes ou scripts.
Il ne faut pas confondre un shell avec un terminal. Un terminal était initialement physiquement un écran et un clavier. Aujourd'hui lorsque l'on parle de terminal on parle d'émulateur de terminal, c'est une catégorie de logiciels permettant de fournir un GUI pour lancer *des* shells (bash, python, zsh, fish, powershell, ruby ...).
Émulateur de terminaux connus : alacritty, Windows Terminal, urxvt, GNOME Terminal, PuTTY.

La confusion est courante car sur Windows historiquement le nom du shell et de l'émulateur de terminal étaient les mêmes (cmd, powershell...), ce n'est plus le cas avec Windows 11 et le Windows Terminal.

2.2 Différents shells Unix

Principaux shells, principales différences, tout pareil osef, focus sur Bash. Abus de langage bash=shell unix

2.3 Rappels Linux essentiels pour le scripting

Supposes qu'ils ont un minimum les bases, donc juste l'essentiel pour scrpiting stdin / stdout return code, droits fichiers, +x, shebang

2.4 Intérets des Shells Unix

Le shell malgré sa syntaxe archaique et ses fonctionnalités limitées vis à vis de langages interprétés à usage général a tout de même encore des avantages.

2.4.1 Actions uniques

Utile lorsque l'on veut faire une tâche relativement simple que quelques fois. Maitriser le shell permet d'être beaucoup plus rapide dans un environnement Unix Exemple : extraire des données textes et les process sommairement cat / sed / cut / tr ...

2.4.2 Manipulation de binaires

L'une des force de Bash par rapport à d'autres langages et sa facilité à manipuler directement des binaires. En python, par exemple, on peut aussi manipuler des binaires, mais on sent clairement que cela est moins pensé pour. Exemple : Utiliser un exécutable propriétaire (import baroc), utiliser l'output d'un script python et l'injecter ailleurs etc

2.4.3 Manipulation de texte

Tout est texte Exemple : lire des logs

2.4.4 Manipulation OS linux

les deux derniers = Manipulation linux pur car linux déjà tout fichiers et texte + binaires unix

Philo kiss, pleins de petits binaires linux avec une seule tache, trop complexe go python

2.5 "flow" de texte

Pipes ! !

2.6 Détails syntaxiques

2.6.1 Redirections et globs (*)

2.6.2 Structures de contrôle

if then else fi while do done until do done for do done case esac
test et [et [[

2.6.3 Variables

2.6.4 Fonctions

2.6.5 Arguments parsing

- vs –

2.6.6 Binaires utiles

Force et efficience des binaires linux cut, cat, echo, grep, tr, sed, xargs, tail, df, ls diff man ! RTFM

2.7 Exercices

Récupérer la liste des pourcentages de remplissages des filesystems

2.7.1 Script

2.7.2 Sans scripts

df | tail -n +2 | tr -s " " | cut -d " " -f 5

Puissance de juste one commande + pipe + redirections

Le Shell POSIX/ISO

L'écriture de script Shell

Activation des commandes POSIX/ISO

Les caractères spéciaux (jockers, échappements, redirection)

Les variables

Les structures de contrôle

3 Le langage Perl - les bases

Prez, utilité de nos jours, spécifité

Présentation de Perl

Les variables scalaires, les tableaux, les opérateurs

Les instructions de contrôle

Les tableaux associatifs (hash)

4 Le langage Ruby - les bases

Présentation de Ruby

Les variables

Les chaînes de caractères

Les structures de contrôle

Les tableaux, les itérateurs - Les hash

5 Le langage Python - les bases

Python est un langage de programmation interprété à usage extrêmement populaire de nos jours. Sa facilité facilité d'apprentissage et de lecture est ce qui a fait sa popularité initiale, aujourd'hui c'est la communauté qui en fait sa force avec les milliers de modules et programmes Python disponibles.

Pour un administrateur système et/ou un devops Python est un langage très attirant, son principal défaut, la potentielle lenteur au runtime, n'est pas un problème dans nos cas d'usage. Python est aussi ce qui est derrière Ansible et permet la création de modules Ansible customs, le plus important outil d'infrastructure as code (IAC) du moment. Il est aussi par défaut installé sur la très grande majorité des distributions Linux.

5.1 Syntaxe

Variables et expressions
Les tableaux, les chaînes de caractères
Les instructions de contrôle
Les dictionnaires (hash)

5.2 Exercices Python

6 Les expressions régulières (RegExp)

regex car tout est texte Importance de grep et sed RegExp en Shell (via grep et sed)
RegExp en Perl (normes)
RegExp en Python (module re mais osef)

7 La modularité en Shell, Perl, Python et Ruby

parler de direnv, de requirements.txt, de venv python, de classes python (osef un peu) y a des trucs pour shell (bpkg) mais pas le but, illogique, shell = spécifique task

Les fonctions => dans les parties syntaxes
Les paquetages=> oui
L'approche objet => syntaxe
Utilisation de bibliothèques externes=> oui

8 La programmation parallèle en Shell, Perl, Python et Ruby

différentes concurrency xargs pipes et parallel pour shell multithreading et async pour python

9 Résoudre des problèmes avec le Shell, Perl, Python et Ruby

Ecrire des scripts d'exploitation (activer une application, les signaux, ...)
Manipuler des fichiers
Faire des calculs
Ecrire des CGI Web
Accéder à des bases de données
Manipuler des fichiers XML (parsing, validation, création)
Créer des applications réseaux TCP/IP

Ptit serveur web Génération de fichier dans un dossier service shell filewatcher qui trigger un truc xargs et des pipes pour multiprocessing perl ou grep pour regex awk pour un truc

9 - AWK : un sous-ensemble POSIX/ISO du langage Perl

10 - Conclusion

Quel outil pour quoi faire ?

Subtitle