R405 : Automatisation des tâches d'administration

Emmanuel Hainry

IUT Nancy Brabois
Réseaux & Télécoms

2023

1. Shell avancé
    - Langage de scripts
    - Redirections
    - Expressions régulières

2. Logiciels et sécurité
    - Stratégies d'Installation
    - Sauvegardes
    - Sécurité et cryptographie

Introduction

- Tâches d'administration
  - Installation, configuration, mises à jour
  - Suivi (veille technologique + Journaux d'événements)
  - Gestion des utilisateurs
  - Sauvegarde
  - Sécurisation
- Automatisation
  - Planifier
  - Programmer (séquence, condition, répétition)
  - Déployer

Descriptif

« Afin d'améliorer l'administration d'un parc informatique ou la surveillance des infrastructures réseaux, le professionnel R&T est amené à développer des scripts permettant le relevé et le traitement automatique des paramètres. »
Mots clés :

- Automatisation
- Scripts
- Expressions régulières
- Journaux d'événements
- Administration système

Langage de scripts

Définition [Scripts]

Un script est un programme *court*, *interprété* qui a un rôle *précis*. Les scripts sont en particulier utilisés dans l'administration système.

Un script est souvent peu maintenable, sale, à usage quasi unique (il pourra être plus simple de refaire le script que de le modifier).
Langages préférés pour le scripting système : **perl**, **python**, ***shell***.

Le langage sh

sh (shell) est à la fois un interpréteur de commande (R108) et un langage de programmation Turing-complet.

- Instructions et Expressions
- Variables et Types
- Structures de contrôle

Instructions

En *sh*, les instructions sont les lignes de commande.
Par exemple

- `cd /tmp`
- `ls`
- `ps -ely`
- `cp /etc/passwd ~`

La séquence d'instructions est obtenue en mettant une instruction par ligne ou en séparant les instructions d'un point-virgule.

Types

Il y a en *sh* un seul type de données : les *chaînes de caractères*.
Pas de booléens, pas de nombres, pas de tableaux.
Un seul opérateur : la concaténation (qui n'a pas de symbole).

- `ab` est la concaténation de `a` et de `b`.

Des commandes permettront de faire des manipulations plus évoluées sur les chaînes telles que

- prendre une sous-chaîne
- trouver un motif
- remplacer un motif

Variables

Identifiant une chaîne de lettres et chiffres commençant par une lettre.

Affectation avec =, sans espaces autour.

Accès en préfixant l'identifiant de $, éventuellement des accolades autour de l'identifiant.

Exemple

- `a1=bonjour`
- `b=12`
- `c=${a1}`
- `echo "$b+2"`
- `e=a$a1b`

Piège : les variables non-initialisées ont une valeur : "".

Expressions

Le résultat d'une instruction peut être utilisé comme une expression en utilisant `$(…)`.
Par exemple

- `processes=$(ps -ely)`
- `ts=$(date "%H:%M")`
- `sh $(curl ${url})`

Conditionnelles

Sans booléens peut-on faire des conditionnelles ?

Conditionnelles

Idée : utiliser le succès ou l'erreur d'une commande.
Par exemple `ls ftest` réussit si `ftest` existe et échoue (*ls : cannot access 'ftest' : No such file or directory*) s'il n'existe pas.

Conditionnelles

Syntaxe

```
if cmd ; then
   act1
else
   act2
fi
```

Sémantique

- Exécuter cmd
- Si réussite, exécuter act1
- Si échec, exécuter act2
- Passer à la suite

```
echo "Bonjour"
if ls /tmp/ftest; then
    touch /tmp/ftest
    echo "/tmp/ftest a été créé"
else
    echo "/tmp/ftest existe déjà"
fi
echo "Dans tous les cas /tmp/ftest existe"
```

Boucle while

Syntaxe

```
act0
while cmd ; do
  act1
done
act2
```

Sémantique

1. Exécuter act0
2. Exécuter cmd
3. Si échec, passer au point 5
4. Si réussite, exécuter act1 puis passer au point 2
5. Exécuter act2

Boucle for

Pas de nombre $\Longrightarrow$ foreach sur les mots
Un mot est une suite de caractères sans caractères d'espacement (espce, passage à la ligne, retour chariot, tabulation, ...)

```
for i in az er ty; do
   echo ${i}
done
```

```
if ls /tmp/blop; then
    cd /tmp/blop
else
    mkdir /tmp/blop
    cd /tmp/blop
fi
for i in $(ls /home); do
    touch "$i"
done
```

Redirections

Mécanisme pour capturer les entrées/sorties des lignes de commande

Sorties

Une commande peut écrire dans le terminal (c'est le cas de *ls*).
On parle de *sortie* de la commande (cf. System.out en java).
Cette sortie peut être redirigée vers un fichier avec > ou >>.

- `ls > fichier`
- `ps > fichier`
- `echo bonjour >> fichier`

Entrée

Une commande peut attendre des entrées (depuis le clavier), à la System.in.
Exemples : `cat`, `sort`.
On peut rediriger un fichier vers l'entrée d'une commande avec <.

- `cat <  /.bashrc`
- `sort < fichier`
- `grep 'x' < fichier > /tmp/x`

Tube

Il est possible de rediriger la sortie d'une commande vers l'entrée d'une autre sans passer par un fichier (on connecte la sortie à l'entrée avec un tube). On utilise pour cela le `|`.

- `ls | less`
- `ls -l | sort`
- `ls | sort | grep 'x'`

filtres

Les commandes comme `sort` qui prennent une entrée et produisent une sortie sont appelées des filtres.
La norme POSIX définit beaucoup de ces commandes par exemple :

- `grep` : filtrer suivant un motif
- `sort` : trier
- `uniq` : supprimer les doublons
- `head` : garder seulement les premières lignes
- `tail` : garder seulement les dernières lignes
- `wc` : compter les lignes, mots et caractères
- `tr` : remplacer lettre par lettre
- `cut` : sélectionner des colonnes
- `sed` : éditer le texte (substitutions, ...)

Expressions régulières

Les *expressions régulières* (regex ou regexp) permettent de définir des motifs textuels.
Ces motifs peuvent être utilisés par exemple par `grep` ou `sed`.
Exemples de motifs :

- les mots qui commencent par un a
- les mots de longueur paire
- les adresses mail
- les mots qui contiennent hainry

Définitions

Définition

Un *alphabet* est un ensemble fini de caractères.
Un *mot* est une suite finie de caractères.
Un *langage* est un ensemble de mots.

Expression Régulière

Définition [regexp]

ε est une regexp, qui capture le mot `""`.
Tout caractère de l'alphabet a, b, ... est une regexp. Elle capture respectivement le mot `"a"`, `"b"`, ...
Étant données deux regexps e_1 et e_2,

- $e_1|e_2$ est une regexp (e_1 ou e_2) qui capture l'union des mots capturés par e_1 et des mots capturés par e_2.
- $e_1 e_2$ est une regexp (concaténation de e_1 et e_2) qui capture l'ensemble des mots $w_1 w_2$ où w_1 est capturé par e_1 et w_2 est capturé par e_2.
- e_1^* est une regexp (e_1 étoile) qui capture l'ensemble des mots de la forme $w_1 w_2 ... w_n$ où chacun des w_i est capturé par e_1 et n un entier naturel éventuellement nul.

Priorités

L'étoile est prioritaire sur la concaténation qui est prioritaire sur le choix.

Ainsi $00|11*$ est équivalent à $(00)|(1(1*))$

En gros, l'étoile est une puissance, la concaténation un produit, le choix une addition.

Exemples

- `(a|b)*` capture l'ensemble des mots composés de *a* et de *b* :
  $\{$"", *a*, *b*, *aa*, *ab*, *ba*, *bb*, *aaa*, *aab*, ...$\}$
- `b(an)*a` capture l'ensemble des mots composés d'un *b* puis d'autant de *an* qu'on veut puis *a* : $\{$*ba*, *bana*, *banana*, *bananana*, ...$\}$
- `00|11*` capture $\{00, 1, 11, 111, 1111, ...\}$
- `0(0|1)1*` capture $\{00, 01, 001, 011, 0011, 0111, 00111, ...\}$
- `(E|e)mmanuel.(Hh)ainry@(loria|univ-lorraine).fr` capture mes adresses mail.

Langage régulier

Définition

L'ensemble des mots capturés par une regexp est le langage *reconnu* (ou *défini*) par cette regexp.

Définition

Un langage est régulier s'il existe une expression régulière qui reconnaît ce langage.

Théorème

Il existe des langages qui ne sont pas réguliers !

Extensions

- `.` est une regexp qui capture chaque caractère. Ainsi, si notre alphabet est $\{a, b, c\}$, la regexp `.` est équivalente à $a|b|c$
- `[a-x]` est une plage de caractères. C'est une regexp qui capture n'importe quel caractère entre a et x. Par exemple
  - `[0-9]` capture les chiffres.
  - `[A-D]` capture les majuscules A, B, C et D.
  - `[a-z]` capture les minuscules.
  - `[A-Za-z]` capture les lettres.

Installation

Quand on installe un programme, on installe

- un exécutable (ex : `firefox.exe`)
- des documentations (ex : pages de man)
- des fichiers de configuration (ex : `debugger.js`)
- des bibliothèques (ex : `sqlite3.dll`)

Les bibliothèques peuvent être partagées entre plusieurs programmes.
Par exemple, sur ma machine, abiword, anki, chromium, gedit, gimp, renpy, toot, virtualbox, ... dépendent de sqlite.
Bibliothèques partagées : sous linux `libxxx.so` ; sous windows `xxx.dll` ; sous macOS `xxx.dylib`.

Systèmes et dépôts d'applications

Comment installe-t-on un logiciel ?

- Sous Windows ?
- Sous Linux ?
- Sous Android ?

Systèmes et dépôts d'applications

Comment installe-t-on un logiciel ?

- Sous Windows ?
  Télécharger un installeur sur le web, Exécuter cet installeur, Cliquer sur OK 8 fois
- Sous Linux ?

- Sous Android ?

Systèmes et dépôts d'applications

Comment installe-t-on un logiciel ?

- Sous Windows ?
  Télécharger un installeur sur le web, Exécuter cet installeur, Cliquer sur OK 8 fois
- Sous Linux ?
  Installer depuis les dépôts de la distribution (`apt install xxxx`)
- Sous Android ?

Systèmes et dépôts d'applications

Comment installe-t-on un logiciel ?

- Sous Windows ?
  Télécharger un installeur sur le web, Exécuter cet installeur, Cliquer sur OK 8 fois
- Sous Linux ?
  Installer depuis les dépôts de la distribution (`apt install xxxx`)
- Sous Android ?
  Installer depuis le dépôt de Google (Play Store)

Systèmes et dépôts d'applications

Comment installe-t-on un logiciel ?

- Sous Windows ?
  Télécharger un installeur sur le web, Exécuter cet installeur, Cliquer sur OK 8 fois
- Sous Linux ?
  Installer depuis les dépôts de la distribution (`apt install xxxx`)
- Sous Android ?
  Installer depuis le dépôt de Google (Play Store)

Il existe des "stores" pour Windows (Windows Store), macOS (Mac App Store), Linux (Flathub,...).
Il existe des dépôts à la Linux pour macOS (macports, homebrew), pour windows (chocolatey).
Il est possible d'utiliser le "modèle" Windows sous Linux, Android, macOS...

Conséquences des stratégies d'installation

Windows Anarchie	Android Isolationisme	Linux Dictature
Installeur place ses versions des bibliothèques dans `C:\System` ⇒ Conflit	Bibliothèques installées dans un dossier propre à chaque programme. Pas de partage ⇒ Duplication	Programmes compilés avec la version de bibliothèque de la distribution
Faille de sécurité Mise à jour d'un des programmes. La bibliothèque est mise à jour pour tous. Faille corrigée, mais certains programmes cassés	Faille de sécurité Chaque développeur doit faire la mise à jour sinon la faille reste présente	Faille de sécurité Le mainteneur fait la mise à jour de la bibliothèque. Les programmes qui en dépendent sont automatiquement recompilés et mis à jour.

Conclusion sur les stratégies d'installation

Le modèle Linux est le plus confortable pour un administrateur :

- pas de conflits de versions
- occupation disque maîtrisée
- mises à jour automatisables

Mais le modèle Android est le plus simple pour un utilisateur :

- pas de conflits de versions
- mises à jour automatiques
- failles de sécurités restreintes

Questions sur les Sauvegardes

Pourquoi	À qui ça sert ?
Quoi	Que doit-on sauvegarder ?
Où	Où seront stockées les sauvegardes ?
Quand	À quelle fréquence faire des sauvegardes ?
Comment	Va-t-on faire de simples copies ou des archives ou... ?

Pourquoi sauvegarder ?

Ne pas perdre de données en cas

- d'erreur utilisateur (Oups, j'ai effacé mon fichier)
- de panne matérielle (Bad blocks)
- de perte, vol, destruction de la machine
- de destruction plus importante (incendie, dégât des eaux...)

Quoi sauvegarder ?

- Sauvegarder les documents utilisateurs
- Sauvegarder les données partagées
  - Documents entreprise
  - Bases de données
- Sauvegarder les configurations des machines

Où, Comment

Idée Copie sur le même disque dur

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)
Pb	Confidentialité (MitM)

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)
Pb	Confidentialité (MitM)
Idée	Utiliser un protocole réseau chiffré (e2ee)

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)
Pb	Confidentialité (MitM)
Idée	Utiliser un protocole réseau chiffré (e2ee)
Pb	Confiance en l'admin nuage (Jennifer Lawrence iCloud)

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)
Pb	Confidentialité (MitM)
Idée	Utiliser un protocole réseau chiffré (e2ee)
Pb	Confiance en l'admin nuage (Jennifer Lawrence iCloud)
Idée	Chiffrer les données avant transfert

Où, Comment

Idée	Copie sur le même disque dur
Pb	ne protège que contre les erreurs utilisateur
Idée	Copie sur un support externe
Pb	ne résoud pas les problèmes d'incendie...
Idée	Sauvegarde distante (nuage)
Pb	Confidentialité (MitM)
Idée	Utiliser un protocole réseau chiffré (e2ee)
Pb	Confiance en l'admin nuage (Jennifer Lawrence iCloud)
Idée	Chiffrer les données avant transfert
Mais	Sauvegarde des clefs de chiffrement...

Quand sauvegarder

- Cas de la perte, vol... $\Rightarrow$ sauvegarde le plus récente possible
- Cas de l'erreur humaine $\Rightarrow$ avoir aussi des sauvegardes anciennes

Donc sauvegardes régulières en conservant un historique.
Par exemple sauvegarde chaque jour, conservation des sauvegardes des 3 derniers jours, plus 4 sauvegardes hebdomadaires plus 11 sauvegardes mensuelles...
Surtout sauvegardes automatiques !

Conclusion sur les sauvegardes

Les sauvegardes doivent être

- distantes,
- chiffrées,
- automatiques,
- régulières,
- vérifiées.

Chiffrement, cryptage, etc.

Utilité de la cryptographie pour le réseau (e2ee), les données chiffrées, la gestion des mots de passe.

- Chiffrement et déchiffrement : codage et décodage d'une information, seuls les destinataires peuvent décoder.
- Décryptage : décodage sans la clef. Décrypter un message revient à briser le chiffrement.
- Signature : preuve de l'identité de l'expéditeur.
- Hashage : une sorte de chiffrement qui ne permet pas de déchiffrement.
- Clef : donnée (mot ou nombre par exemple) utilisée par l'algorithme de chiffrement pour chiffrer ou déchiffrer.

Example (Chiffre de Vigenère)

Chaque lettre est remplacée par la lettre +n. (A par B, B par C... si n=1).
n est la clef de chiffrement.
On sait décrypter un message.

Cryptographie symétrique et asymétrique

- Cryptographie symétrique : la clef de chiffrement est aussi la clef utilisée pour déchiffrer.
  C'est le cas classique.
  Implique de partager la clef avec le destinataire.
- Cryptographie asymétrique : une clef pour chiffrer et une autre pour déchiffrer.
  Si la clef de chiffrement est publique et la clef de déchiffrement secrète, tout le monde peut m'envoyer un message que je serai le seul à pouvoir déchiffrer.
  Implémenté pour la première fois en 1977 (RSA).

- Message chiffré $\Rightarrow$ crypto asymétrique
- Signature $\Rightarrow$ crypto asymétrique
- Sauvegarde $\Rightarrow$ crypto symétrique

Gestion des mots de passe

Pouvoir déchiffrer les mots de passe serait une faille de sécurité !

Idée du hashage : codage sans décodage possible.

Vérifier un mot de passe = vérifier les hashs.

Implémentation possible : crypto asymétrique en jetant la clef secrête.