Système d'exploitation
Arel et Google

Florent Devin

5 décembre 2011

1 Introduction

1.1 Présentation

L'objet de ce Tp, et du suivant est de faire un script shell pour vous permettre d'importer automatiquement votre calendrier AREL dans votre agenda Google. Comme cela demande quelques traitements, nous allons décomposer en plusieurs phases. Dans une première phase, vous allez vous occuper de transformer une chaîne texte (produite par AREL), en une chaîne texte utilisable par *Google Command Line*. Dans un second temps, nous aborderons l'extraction des données depuis AREL ainsi que le transfert (et éventuellement un traitement) dans votre agenda Google.

Listing 1 – Extrait d'AREL

```
<plannings>
      <demandeur id="0"></demandeur>
      <utilisateur id="1574" nom="Devin" prenom="Florent"></utilisateur>
      <emploiDuTemps noSemaine="49" anneeCivile="2011"></emploiDuTemps>
            <creneau id="39930" annee="2011"
                              mois="12" jourMois="1"
                              heureDebut="08h30" heureFin="11h00"
                              idProf="1574" prof="fd"
                              salle="E110" groupe="A"
                              salleId="114" groupeId="1985"
                              visio="null">
    <prog sessionId="72"
                                    id="23"
                                    objectif="Ingénierie_en_Cloud_Computing" />
                        <rel id="9393" annee="2011"
        libelle="Ingénierie_du_Cloud_Computing" type="matière"/>
            </creneau>
            <creneau id="39926" annee="2011"
                              mois="11" jourMois="30"
                              heureDebut="13h30" heureFin="16h00"
                              idProf="1574" prof="fd"
                              salle="E110" groupe="A"
                              salleId="114" groupeId="1985"
                              visio="null">
    <prog sessionId="72"
                                    id="23"
                                    objectif="Ingénierie_en_Cloud_Computing" />
                        <rel id="10497" annee="2011"
        libelle="Enjeux_et_perspectives_du_Cloud_Computing" type="matière" />
            </creneau>
            <creneau id="39875" annee="2011"
                              mois="11" jourMois="30"
                              heureDebut="09h45" heureFin="11h15"
                              idProf="1574" prof="fd"
```

```
                                    salle="E215" groupe="A"
                                    salleId="107" groupeId="1985"
                                    visio="null">
      <prog sessionId="62"
                                          id="5"
                                          objectif="ING1" />
                          <rel id="2335" annee="2011"
            libelle="Système d'exploitation" type="matière" />
                    </creneau>
</plannings>
```

Listing 2 – Pour l'import dans Google en CSV

```
Subject, Start Date, Start Time, End Time, End Date
Ingénierie du Cloud Computing (E110),01/12/2011,08:30,11:00,01/12/2011
Enjeux et perspectives du Cloud Computing (E110),30/11/2011,13:30,16:00,30/11/2011
Système d'exploitation (E215),30/11/2011,09:45,11:15,30/11/2011
```

1.2 Contexte

Dans ce Tp, il est question de transformer un fichier qui ressemble au listing 1 en un fichier qui ressemble au listing 2. Bien que le listing 1 représente un fichier XML, nous n'allons pas aborder les techniques habituelles de traitement de ce type de fichier [1].

Il y a beaucoup de moyen de transformer ce fichier, nous pourrions utiliser un programme externe qui facilite le traitement du style `xmllint`. Ceci dit la maîtrise d'un tel outil est quelque peu délicat, et requiert des compétences que vous n'avez pas forcement. On pourrait essayer de faire un traitement manuel : on observe la façon dont est écrit le fichier original, et l'on applique un traitement spécifique. Si cette méthode est souvent une méthode utilisée, c'est dans ce cas une mauvaise idée.

En effet, quand vous extrayez des données depuis un site web, rien ne vous indique que le concepteur du site web ne va pas changer la façon d'écrire le fichier, ou du moins l'ordre des balises. Nous allons donc essayer de traiter ce fichier de la façon la plus générique possible.

Nous pourrions aussi utiliser les commandes `awk` et `sed`, mais ces commandes ne sont pas indispensables pour le TP.

1.3 Outillage

```
Bonjour;tout;le;monde
Bonjour tout le monde
Bonjour;;;tout  le   monde
```

Code 1: Fichier : toto

1. Nous aborderons cela au second semestre, vous pourriez alors réécrire ce script

1.3.1 IFS

IFS [2] (ou *Internal Field Separator*) est une variable système qui spécifie quels sont les séparateurs en vigueur. Normalement sur votre système vous devriez avoir l'espace, la tabulation et aussi le retour à la ligne. Unix fait un usage régulier de cette variable. Pour preuve, créez un fichier identique à celui sur le code 1 que vous nommerez par exemple `toto`. Puis tapez la même séquence que celle sur le code 2 et observez attentivement le résultat. Cela devrait vous aider à comprendre le fonctionnement de IFS.

```
OLDIFS=${IFS}
for i in `cat toto`; do echo "Valeur de i : ${i}";done
IFS=;
for i in `cat toto`; do echo "Valeur de i : ${i}";done
IFS=\;
for i in `cat toto`; do echo "Valeur de i : ${i}";done
IFS=${OLDIFS}
```

Code 2: Commandes à exécuter sur toto

Explication sur la ligne :

```
IFS=\;
```

Nous sommes obligés de mettre un \ devant le ;, car le caractère ; est un caractère qui marque la fin d'une instruction. Il faut donc mettre un \ devant, en jargon on appelle cela l'échapper.

1.3.2 Tableaux

Le shell est capable de traiter des tableaux ! Vous trouverez sur le code 3 un ensemble de commandes indispensables pour "survivre" à l'utilisation des tableaux. Cependant le shell gère les tableaux à sa façon :

- Les tableaux n'ont pas de taille prédéfinie
- Les indices d'un tableau ne sont pas forcement consécutifs, peu recommandé, mais possible
- On peut rajouter des éléments en tête du tableau (ou en fin) (si les indices sont consécutifs)
- Les tableaux ne sont "que" mono-dimensionnel

Lors de la création et de l'utilisation de tableaux, il vaut mieux éviter de manipuler des tableaux à trous. Le problème c'est que parfois, lors de la création du tableau, nous n'avons pas le choix. Vous pouvez alors soit traiter le tableau en connaissance de cause, soit renuméroter les indices. Le plus "amusant" c'est

2. N'ayez pas peur, je ne parle pas de fractales ;-)

```
#Pour connaitre tous les tableaux existants et leurs valeurs
declare -a

#Pour declarer un tableau
declare -a TAB=(Bonjour tout le monde)
TAB=( Bonjour tout le monde )
TAB=( $(echo "Bonjour tout le monde") )

#Pour declarer un tableau en lecture seule
declare -ai TAB=(Bonjour tout le monde)

#Manipulation des tableaux
echo ${TAB[1]}                =>   tout
echo ${#TAB[*]}               =>   4 (nb element du tableau)
echo $TAB                     =>   Bonjour
echo ${TAB[89]}               =>   #n'affiche rien
echo ${TAB[1*2]}              =>   le
echo ${TAB[*]}                =>   Bonjour tout le monde

#liste des indices du tableau
echo ${!TAB[*]}               =>   0 1 2 3

#Ajout d'un element
TAB[7]=ICC                    =>  # ajoute a la position 7 !!!

#Ajout en tete et renumerotation des indices (effet de bord)
TAB=( Hello ${TAB[*]} )

#Ajout en fin
TAB[${#TAB[*]}]=Bye

#Renumerotation des indices
TAB=( ${TAB[*]} )
```

Code 3: Commandes utiles pour manipuler les tableaux

que les tableaux utilisent l'IFS pour délimiter leurs valeurs. Après avoir compris les commandes du code 3, essayez de taper la commande suivante :

```
IFS=o;TAB=( $(echo "Bonjour tout le monde") );declare -a
```

Que remarquez vous ?

2 Votre travail

Réalisez un script qui permet de traiter un fichier comme représenté sur le listing 1, en quelque chose qui ressemble au listing 2. Pensez que ce script sera réutilisé la semaine prochaine.