Séance 1: Scripts et Shell

IUT SD Aurillac

Objectifs:

— Gestion des paramètres en ligne de commande | — Rappels sur Shell

L'automatisation informatique consiste à utiliser des logiciels et des scripts pour créer des processus reproductibles dans le but de réduire les interventions humaines. L'idée est de mettre en place un script shell combinant des programmes de langages divers et qui gère automatiquement un maximum d'étapes (récupération du fichier de données, nettoyage, enregistrement des données temporaires, analyse, production des visualisation,...), le tout avec le moins d'intervention du programmeur ou du client sur le process, une fois celui-ci finalisé.

Le premier outil nécessaire pour cela est l'utilisation de paramètres lors de l'exécution en ligne de commande d'un programme, que ce soit Python, R, Java ou autre. Ces paramètres peuvent être référencés et utilisés dans le fichier lui-même et permettent d'éviter d'avoir à modifier le fichier manuellement, ou de placer un input qui oblige une intervention humaine. Nous allons voir comment se servir de ces paramètres.

— Python

En Python, la syntaxe pour ajouter des paramètres est la suivante : `python3 test.py arg1 arg2`
Pour récupérer ces paramètres dans le script, il faut importer le module `sys`. ils sont stockés dans la variable `sys.argv` sous forme de chaine de caractères.

— R

Tout d'abord en R, pour executer un fichier en ligne de commande, il suffit d'entrer la commande `R --vanilla <test.r`.
Pour ajouter des paramètres à cette commande, il suffit de la modifier ainsi : `R --vanilla --args arg1 arg2 <test.r`
On peut ajouter un nombre quelconque d'arguments après l'option `--args`.
Pour récupérer ces paramètres dans le script, il suffit d'utiliser la fonction commandArgs : `tab<-commandArgs(TRUE)`. A noter que ces arguments sont automatiquement de type character, et qu'il faudra donc les convertir pour les utiliser comme d'autres types (notamment pour des nombres).

— Bonus VCOD : Java
En java, la syntaxe pour ajouter des paramètres est la suivante : `java Test arg1 arg2`
Pour accéder à ces paramètres, c'est très simple. Ils sont stockés dans le tableau `args` qui est en paramètre de la méthode `main` depuis notre premier programme java.

Exercice 1 (, ⋆)
Ecrire un script (python ou r ou java) prenant en paramètre trois entiers et renvoyant vrai si les trois entiers correspondent aux longueurs d'un triangle, et faux sinon. □

Exercice 2 (, ⋆)
Ecrire un script (python ou r ou java) prenant en paramètre un nom de fichier contenant des paires(noms d'étudiants, notes) et créant un nouveau fichier ajoutant une appréciation ("très bien" si la note est entre 16 et 20 , "bien" lorsqu'elle est entre 14 et 16 , "assez bien" si la note est entre 12 et 14 , "moyen" si la note est entre 10 et 12 ,"insuffisant" si la note est inférieure à 10.). □

Intermède : le shebang

Le shebang est une ligne à ajouter au début d'un fichier de code, qui permet à l'ordinateur de savoir quelle est la commande nécessaire pour exécuter ce fichier. Cela permet un allègement de l'écriture des commandes dans la console (ou dans les scripts shell).

— Python
En Python, il faut écrire la ligne `#!/usr/bin/env python3`.
Le fichier peut alors être exécuté (si vous avez les droits d'exécution dessus) simplement avec la commande `./test.py`.

— R
En R, il faut écrire la ligne `#!/usr/bin/env Rscript`.
Le fichier peut alors être exécuté (si vous avez les droits d'exécution dessus) simplement avec la commande `./test.r`.

Le Shell est le langage de programmation utilisé par le terminal. Il en existe différentes variations selon l'OS et la version.

Nous avons déjà vu depuis le début de l'année dernière un grand nombre de commandes (`ls,cd,rm, touch ,...`) mais nous n'avons pas vraiment posé les bases du langage :

— On peut bien sûr écrire des scripts shell directement dans la console, ou bien dans des fichiers .sh, qui sont ensuite exécutés dans la console. (le shebang pour shell est `#!/bin/bash`)

— `echo` est la version shell de print. Cette commande a aussi l'avantage de pouvoir écrire directement dans un fichier avec la syntaxe `echo text > fichier.txt`

— la gestion des variables en shell est un peu particulière. Pour initialiser ou modifier une variable, il suffit de faire $a = 5$. Pour s'en servir en revanche, il faut que le nom de la variable soit précédé de $, par exemple $a = \$((\$a + 1))$.
Attention, si vous souhaitez stocker le résultat d'une commande dans une variable, la syntaxe est un peu différente. Il faut entourer cette commande de $\$()$. Par exemple, avec la commande `date` qui génère la date du jour, on écrit $d = \$(date)$ pour stocker le résultat de date dans la variable d

— La syntaxe des conditions est un peu particulière :

```
if  [ condition ]
then
  command
else
  command
fi
```

A noter que pour les conditions, il existe des opérateurs qui sont différents pour les différents types. Vous trouverez une liste non exhaustive ici : https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html

— Voici la syntaxe des boucles for, proche de celles de python :

```
for var in liste
do
  command
done
```

L'idée est de combiner cette structure avec des commandes générant des listes (comme ls, seq ou cat par exemple). Par exemple, pour boucler sur les fichiers py présents dans le dossier d'exécution :

```
for var in $(ls *.py)
do
  echo $var
done
```

— Lorsqu'on exécute un script en ligne de commande, on peut aussi lui passer des paramètres. Ceux-ci sont référencés dans le code par $\$1, \$2, \$3, ...$ dans leur ordre d'ajout.

Exercice 3 (, $\star$)
Écrire un script shell qui attend jusqu'à quatre arguments sur la ligne de commande, et les affiche en ordre inverse.

Les arguments au delà du quatrième seront ignorés.

□

Exercice 4 (, $\star$)
Écrire un script qui crée 5 fichiers $fic1.txt$ à $fic5.txt$, contenant respectivement les valeurs 1 à 5, dans un répertoire passé en paramètre.

Si ce répertoire n'existe pas, il doit être créé. Si un des cinq fichiers existe déjà, un message d'erreur doit être affiché.

□

Exercice 5 (, $\star$)
Écrire un script prenant un répertoire et trois entiers en paramètre, et exécutant tous les fichiers python contenus dans ce répertoire sur ces trois entiers.

□

Exercice 6 (, $\star$)
Ecrire un script prenant trois paramètres : le premier désigne le nom du fichier dont on veut copier le contenu et les 2ème et 3ème arguments sont les noms des fichiers vers lesquels on veut faire la copie du premier. □

Exercice 7 (, $\star\star$)
Créer un script qui doit calculer le nombre de fichiers standard, de sous-répertoires, et d'exécutables d'un répertoire quelconque qui sera donné en paramètre. □

Exercice 8 (, $\star$)
Ecrire un script prenant en paramètre une chaine de caractère représentant une extension de fichier(jpg,txt,py,...) et ajoutant au nom de tous les fichiers dans le répertoire courant ayant cette extension la date du jour. □

Exercice 9 (, $\star$)
Ecrire un script qui retire tous les commentaires (mais pas le shebang) d'un script python pris en paramètre et enregistre cette version du script dans un nouveau fichier. □