Programmation en Scilab

1. Introduction

Une succession de d'instructions est appelée un **script**. On peut les saisir directement dans la console **Scilab**, mais ceci a de nombreux inconvénients, car à chaque modification du script (même mineure), il faut le réécrire en entier. De plus, il est difficile de le sauvegarder pour une utilisation ultérieure. C'est pour ces raisons qu'on utilise systématiquement l'éditeur intégré **Scinotes**.

2. L'éditeur SciNotes

À partir de la fenêtre principale de **Scilab**, l'onglet **Applications** permet de lancer l'éditeur **SciNotes**.

- Le fichier doit être sauvegardé avec le suffixe **.sce** pour indiquer au système que c'est un fichier **Scilab**.
- Pour assurer une compatibilité entre les divers systèmes d'exploitations, les noms de fichiers doivent être en un seul mot (pas d'espace), sans accent, sans tiret et sans point (sauf le **.sce**). Par contre, on peut utiliser le caractère souligné _ pour un nom de fichier (il remplace le caractère espace).
- Pour sauvegarder à partir de **Scinotes**, on utilise l'onglet **Fichier** puis **Enregistrer sous** si on veut donner un nouveau nom au fichier, ou **Enregistrer** si ce n'est pas nécessaire.
- Le répertoire, appelé **répertoire courant**, dans lequel on travaille, est donné par la commande **pwd** ou via l'onglet **Fichier** de la fenêtre principale. Il se change via la commande **chdir** ou via l'onglet **Fichier**.
- Il est conseillé de créer un répertoire **Scilab** puis un sous- répertoire par séance de **TP,** pour sauvegarder à chaque fois tous les fichiers.
- Pour exécuter un programme il faut le sauvegarder dans le répertoire courant, puis à partir de l'éditeur Scinotes, on clique sur l'onglet **Exécuter**, et on choisit **Enregistrer et Exécuter** Cette manipulation exécute le programme dans la **console Scilab**. Dans SciNotes, la touche **Entrée** ne sert donc plus à exécuter les commandes, mais simplement à passer à la ligne suivante (comme dans n'importe quel éditeur de texte).
- Pour corriger/modifier un programme il suffit d'éditer le fichier avec **SciNotes**. Si le fichier a été fermé, ou peut l'ouvrir à nouveau avec l'onglet **Fichier** puis **Ouvrir**.
- Lorsque l'exécution d'un programme ne s'arrête pas, on peut forcer l'arrêt en pressant simultanément les touches **Ctrl** et **C**. Si cela ne fonctionne pas, il ne reste qu'à fermer **Scilab.**
- On peut ajouter des commentaires pour se rafraîchir la mémoire, ou pour expliquer le programme à une tierce personne. Il suffit de placer // en début de ligne : **elle ne sera alors pas exécutée par Scilab**. Cela permet aussi de chercher les erreurs : on empêche l'exécution de certaines lignes pour trouver celle(s) qui cause(nt) une erreur.
- Les variables utilisées dans un programme sont **globales** : on peut les utiliser et les modifier aussi bien dans la console que dans un programme.

3. Lecture

La commande **input** permet de demander à l'utilisateur d'un programme de fournir des données.

a) Lecture d'une variable de type numérique, de type logique(booléenne) ou une expression (Arithmétique ou booléenne)

La syntaxe est : **var = input(' une phrase ')**. Ou **var = input(" une phrase ")**

- La phrase « **une phrase »** est affichée et **SCILAB** attend que l'utilisateur saisisse une donnée au clavier.
- Cette donnée peut être une valeur numérique, une valeur booléenne ou une instruction **SCILAB (arithmétique ou logique)**.
  Un retour chariot provoque la fin de la saisie.
- Une valeur numérique ou logique est directement affectée à la variable **var** tandis qu'une **instruction SCILAB** est évaluée et le résultat est affecté à la variable **var**.

Exemples

```
1)  -->A = input ('Entrez un nombre entier : ')  // Lecture d'un numérique
Entrez un nombre entier : 56
 A  =
  56.

2)  -->A = input ('Entrez un nombre entier : '); // Lecture d'un numérique sans echo
Entrez un nombre entier : 57
 A  =
  57.

3)  -->LOGI = input ("Entrez une valeur logique : ") // Lecture d'un booléen
Entrez une valeur logique : %t
 LOGI  =
  T

4)  -->E = input ('Entrez une expression arithmétique : ') // Lecture d'une expression
    arithmétique
Entrez une expression arithmétique : 2*5+6-7
 A  =
  9.

5)  -->EB = input ('Entrez une expression logique : ') // Lecture d'une expression
    arithmétique
Entrez une expression logique : 23>90
 EB  =
  F
```

6) -->B = input ("Entrez un vecteur ligne : ") // Lecture d'un vecteur (énumération)
Entrez un vecteur ligne : [3 5 1 7]
B =
3. 5. 1. 7.

7) -->B = input ('Entrez un vecteur ligne : ') // Lecture d'un vecteur (description)
Entrez un vecteur ligne : [1:3]
B =
1. 2. 3.

8) -->M1 = input ("Entrez une matrice : ") // Lecture d'une matrice (énumération)
Entrez une matrice : [1 2 3;0 9 8;5 6 7]
B =
1. 2. 3.
0. 9. 8.
5. 6. 7.

9) -->M2 = input ("Entrez une matrice : ") // Lecture d'une matrice (Description)
Entrez une matrice : [5:5:20;90:-4:78]
M2 =
5. 10. 15. 20.
90. 86. 82. 78.

b) Lecture d'une variable de type chaîne de caractère

Si l'on souhaite saisir une réponse de type chaîne de caractères on utilise la syntaxe :
var = input(' une phrase ','s') **ou** **var = input(' une phrase ','string')**.

10) -->text=input(' Tapez un texte : ','s') // lecture d'une variable de type chaîne de caractère
Tapez un texte : Bonjour
text =
Bonjour

11) -->text=input(' Tapez un texte : ','string')
Tapez un texte : 34
text =
34 («**34** » est la chaîne de caractère **'34**' et non la valeur **34**)

4. Ecriture ou Affichage

Pour qu'un programme affiche la valeur d'une variable ou du texte, on utilise la commande **disp**.

Exemples

1) Afficher la valeur d'une variable (numérique ou booléenne)

```
--> v=[6:3:27];  // Création d'un vecteur sans affichage
--> v // Affichage du vecteur v
 v  =
   6.  9.  12.  15.  18.  21.  24.  27.
--> disp(v) // Affichage du vecteur v avec disp, voyez la différence !
   6.  9.  12.  15.  18.  21.  24.  27.
```

- Le principe est le même pour tous type de variable (numérique, booléenne, scalaire, vecteur ou matrice) !

2) Afficher du texte

```
--> disp('ALGERIE, mon beau pays')
 ALGERIE, mon beau pays
```

3) Afficher plusieurs objets

On peut afficher plusieurs objets avec la même commande :
disp(objetn**,....,objet**2**,objet**1**)**.

– Attention il faut les objets dans l'ordre inverse de celui souhaité pour l'affichage !

Exemples

```
--> disp(1,2,3,4,5)
   5.
   4.
   3.
   2.
   1.

--> disp('Cinq','Quatre','Trois','Deux','Un')
 Un
 Deux
 Trois
 Quatre
 Cinq
```

5. Exemple de script