Présentation du langage J

par Chris Burke
(Strand Software Systems Inc.)

I - LA PRÉSENTATION

Chris a présenté J comme une version moderne d'APL, écrite par Ken Iverson et Roger Hui en un système complètement redéveloppé à partir de rien en 1989. Les premières versions furent distribuées en 1990 et la commercialisation débuta en 1994. A la question « Pourquoi J ? » Chris répondait en s'appuyant sur un transparent où on lisait:

J garde les meilleurs caractéristiques d'APL
J résout les problèmes d'APL
- le problème du jeu de caractères
- les fautes du langage
- un environnement de développement impropre
- l'isolation par rapport aux autres systèmes
- la difficulté d'y ajouter des nouveautés

suivi d'un autre comparant J et APL: Plus simple et plus cohérent Un environnement standard de programmation

- les programmes sont développés dans des fichiers « texte » ordinaires
-Utilise n'importe quel éditeur et système de commandes
- Accès aisé par les autres logiciels
Des nouveautés puissantes
- amélioration des primitives APL
- rang des fonctions
- programmation fonctionnelle
- structures de contrôle et espaces locaux

Chris a annoncé la version J3 aux performances supérieures à APL incluant les commandes WIN95 et le support OLE2 (J comme objet OLE, accès OLE à Excel, Word...)

Enfin Chris a insisté sur l'intérêt qu'il y a à envisager **J** comme un serveur de calcul pour Visual Basic, Delphi, Excel, etc.

II - LES EXEMPLES

La présentation du langage **J** a eu pour but de donner aux APListes présents une idée des immenses possibilités du langage à partir de deux exemples simples : trier les mots d'une chaîne de caractères et concaténer une matrice à un vecteur.

1 - Tri de mots

L'affectation se fait en J à l'aide du signe =.

```
texte .='jules marie henri anne gilbert'
```

La fonction ;: prends les éléments (token) d'une chaîne de caractères et les enclôt.

```
lesmots =.  ;:      NB. définition d'une fonction

lesmots texte
```

jules	marie	henri	anne	gilbert

Le nombre d'éléments du résultat est cinq (5 mots enclos).

Le tri est une fonction dyadique /: telle que x /: y trie x selon les indices donnés par le tri monadique /: y. Une fourche avec «déclore» (>) fait le reste sachant que la composition par fourche (g h) x signifie x g (h x).

```
tri=: /: >

 tri les mots texte
```

anne	gilbert	henri	jules	marie

On voit ainsi la différence entre (tri lesmots texte) et (tri texte)

```
      tri texte
  aaaabeeeghijlmmnnrrrrstyy
```

Définissons la conjonction «puissance» telle que (f puissance n) applique n fois la fonction f et définissons la conjonction «inverse» qui est «puissance moins un»:

```
      puissance =. ^:
      inverse =. puissance _1
```

«inverse» appliqué à «lesmots» crée la fonction qui déclot et remet en liste les éléments:

```
      (lesmots inverse) lesmots texte
jules marie henri anne gilbert
```

On peut créer la fonction correspondante:

```
      enliste =. lesmots inverse

      enliste lesmots texte
jules marie henri anne gilbert
```

On peut, à l'aide d'une conjonction (opérateur), effectuer une fonction f1 **sous** une autre fonction f2, en ce sens qu'on applique d'abord f2, puis f1 au résultat, puis l'inverse de f2 au résultat final. Par exemple, raccorder deux fils électriques, c'est : dénuder les fils (f2), torsader le cuivre (f1) et remettre l'isolant (inverse de f2). On a torsadé le cuivre sous sa mise à nu : on a effectué f1 sous f2. Cette conjonction se note &. en **J**.

```
      sous=: &.
      trilesmots=. tri sous lesmots

      trilesmots texte
anne gilbert henri jules marie
```

En synthèse, on montre ci-dessous comment on définit un tri alphabétique sur un vecteur de caractères et comment on le transforme en un tri sur les mots.

```
      tri=. /: >
      trilesmots=. tri &. ;:

      texte .='jules marie henri anne gilbert'

      tri texte
 aaaabeeeghijlmmnnrrrrstyy

      trilesmots texte
anne gilbert henri jules marie
```

Incidemment, la conjonction « puissance » (introduite pour créer la conjonction « inverse ») peut être aussi exploitée sur des fonctions arithmétiques, y compris avec des arguments vectoriels. **J** offre la fonction « double » (notée +:) qui rend de nombreux services.

```
      double=. +:

      double 1
2
      double double 1
4
      double double double 1
8
      double puissance (3) 1
8

      double puissance (0) 1
1
      double puissance (0 1 2 3 4 5 6 7 8 9) 1
```

```
1 2 4 8 16 32 64 128 256 512

      double puissance(i. 10) 1 (i. est iota en 0←OI□)
1 2 4 8 16 32 64 128 256 512
```

(Puissance moins 1) ou (puissance inverse) donne l'inverse de « double » qui est la fonction « moitié ».

```
      double puissance (_1) 1
0.5
      double inverse 1
0.5
```

« moitié » existe aussi et se note (-:) en ***J***.

```
      -: 1
0.5
```

2 - Concaténation d'une matrice et d'un vecteur

Prenons une matrice x et un vecteur y et concaténons les de toutes les manières possibles avec la conjonction de rang.

```
   x                y
1 2 3            10 20 30
4 5 6
7 8 9
```

La matrice x est un objet de rang 2 qui peut être vu comme tel ou, au rang 1 comme un vecteur de lignes ou, au rang 0, comme un tableau de scalaires. Le vecteur y est un objet de rang 1 qui peut être vu comme tel ou, au rang 0, comme une liste de scalaires.

Ces trois vues pour x et ces deux vues pour y peuvent conduisent à six concaténations différentes. En utilisant l'opérateur de rang **"**, on peut en effet avoir les six concaténations présentées dans le tableau ci-dessous.

```
x,"2 1 y
x,"1 1 y
x,"0 1 y

x,"2 0 y
x,"1 0 y
x,"0 0 y
```

Les six autres possibilités de base se déduisent en prenant l'ordre y,x au lieu de x,y. Enfin, on démultiplie encore les possibilité en effectuant ses concaténations **sous** (&.) «transpose» (|:).

```
x,"2 1 y
1  2  3
4  5  6
7  8  9
10 20 30
```

```
x,"1 1 y
1 2 3 10 2
4 5 6 10 2
7 8 9 10 2
```

```
x,"0 1
 1 10 20
 2 10 20
 3 10 20

 4 10 20
 5 10 20
 6 10 20

 7 10 20
 8 10 20
 9 10 20
```

```
x,"2 0 y
1  2  3
4  5  6
7  8  9
10 10 10

1  2  3
4  5  6
7  8  9
20 20 20


1  2  3
4  5  6
7  8  9
30 30 30
```

```
x,"1 0 y
1 2 3 10
4 5 6 20
7 8 9 30
```

```
x,"0 0
1 10
2 10
3 10

4 20
5 20
6 20

7 30
8 30
9 30
```

Chris Burke en est resté à ces exemples et a ensuite répondu aux questions de l'auditoire touchant essentiellement à la forme tacite de l'écriture des fonctions et aux interfaçages avec WINDOWS.