Quelques notions d'IDL

Michel Fioc

1 Programme principal, procédures, fonctions

Les fichiers les contenant ont nécessairement le suffixe **.pro**.

- Le format du programme principal est
  ...
  ...
  end
- Le format d'une procédure est
  pro *nom_procédure*, *variable*
  ...
  return
  end
  Elle est appelée par *nom_procédure*, *variable* (ou *nom_procédure*, *valeur* si la variable n'est pas modifiée dans la procédure).
- Le format d'une fonction est
  function *nom_fonction*, *variable1*, *variable2*
  ...
  return, *valeur_fonction*
  end
  Elle est appelée par ... **=** *nom_fonction***(***variable1***,***variable2***)**.

Si le fichier contenant le programme principal contient également des procédures ou des fonctions, celles-ci doivent être placées avant.

Si une procédure ou une fonction est contenue dans un fichier séparé, il est préférable de donner au fichier le même nom, avec le suffixe **.pro**, que la procédure ou fonction.

Les procédures et fonctions doivent être compilées avant le programme principal.

- **.run** *nom1*, *nom2* : Compile *nom1***.pro** et *nom2***.pro**. Si le programme principal est contenu dans l'un de ces fichiers, il est exécuté.
- **.rnew** *nom* : Même chose que **.run**, mais détruit en plus toutes les variables.
- **.compile** *nom* : Compile sans exécuter.
- **.go** *nom* : Exécute un programme principal déjà compilé.

N.B. : On peut se passer d'un programme principal. On peut notamment exécuter une procédure directement en tapant son nom.

2 Instructions de contrôle

- **for** *variable* **=** *valeur1*, *valeur2* **do** : Boucle de *valeur1* à *valeur2* sur *variable* pour une seule instruction.
- **for** *variable* **=** *valeur1*, *valeur2*, *valeur3* **do begin**
  ...
  endfor : Idem par pas de *valeur3* pour plusieurs instructions.
- **goto,** *identificateur*
  ...
  *identificateur***:** ... : Renvoi à la ligne précédée de *identificateur***:**.
- **if** ... **then** ..., ou **if** ... **then begin**
  ...
  endif : Si ..., alors ...

- **if** ... **then** ... **else** ..., ou **if** ... **then begin**
  ...
  endif else begin
  ...
  endelse : Si ..., alors ... ; sinon ...
- **repeat** ... **until** ..., ou **repeat begin**
  ...
  endrep until ... : Répéter ... jusqu'à ce que ...
- **while** ... **do**, ou **while** ... **do begin**
  ...
  endwhile : Tant que ..., faire ...
- **case** *variable* **of**
  *valeur1***:** *instruction1*
  *valeur2***: begin**
  ...
  end
  else: ...
  endcase : Si *variable* vaut *valeur1*, alors *instruction1* ; si elle vaut *valeur2*, ... ; sinon ...
- **stop,** *expression* : Le programme s'arrête et affiche *expression* (si *expression* est présent).

3 Fichiers

- **get_lun,** *unité* : Écrit un numéro d'unité libre (entier entre 1 et 99) dans la variable *unité*.
- **openr,** *unité***,** *'fichier'* : Ouvre le fichier *fichier* en lecture et lui associe le numéro *unité*. **openw** pour écrire un nouveau fichier, **openu** pour modifier un fichier existant.
- **close,** *unité* : Ferme le fichier numéro *unité*.
- **printf,** *unité***,** *variable1***,** *variable2* : Écrit les variables *variable1* et *variable2* dans le fichier numéro *unité*.
- **readf,** *unité***,** *variable* : Lit la variable *variable*.
- **print,** *variable* : Écrit la variable *variable* à l'écran.
- **read,** *variable* : Lit la variable *variable* tapée au clavier.
  Note : On peut faire **readf,** *unité***,** *tableau*, mais pas **readf,** *unité***,** *tableau***[***indice***]** (car IDL passe les tableaux complets par adresse, mais les éléments de tableaux par valeur). Pour cela, il faut faire **readf,** *unité***,** *scalaire*, puis *tableau***[***indice***] =** *scalaire*.
- **eof(***unité***)** : Test de fin du fichier numéro *unité*.

4 Graphiques

- **set_plot, 'ps'**
  device, file = *'fichier.ps'*
  ...
  device, /close : Représente les graphiques dans un fichier postscript de nom *fichier.ps*. Autres possibilités (au lieu de **'ps'**) : **'hp'**, **'null'**, **'win'**, **'x'**...
- **cursor,** *x***,** *y* : Donne la position du curseur.
- **erase** : Efface la fenêtre.
- **window,** *i* : Crée une fenêtre portant le numéro *i*.
- **wsel,** *i* : Sélectionne la fenêtre numéro *i* pour les graphiques suivants.
- **plot,** *x***,** *y* : Trace la variable *y* en fonction de *x* (*x* et *y* sont des vecteurs).
- **plot_io,** *x***,** *y***, title =** *'...'***, xtitle =** *'...'***, ytitle =** *'...'* : Idem avec un axe logarithmique pour *y*, le titre du graphique et les légendes des axes.
- **plot_oi,** *x***,** *y***, linestyle =** *i* : Idem avec un axe logarithmique pour *x*. Le type de courbe est le numéro *i* (p. ex. tirets = 2).

- **plot_oo,** x**,** y**, thick =** i : Idem avec un axe logarithmique pour x et y. La largeur de la courbe est de i.
- **plot,** x**,** y**, psym =** i**, color =** j : Trace des points (avec le symbole numéro i, la couleur numéro j) plutôt qu'une courbe continue.
- **plot,** x**,** y**, xrange = [**x_1**,** x_2**], xstyle =** 1 : Les abscisses vont de x_1 à x_2 (idem avec **yrange** pour les ordonnées). Sans le **xstyle =** 1, le **xrange** n'a qu'une valeur indicative.
- **oplot,** x'**,** y' : Trace la variable y' en fonction de x' sur le même graphique.
- **xyout,** x**,** y**,** *texte* : Écrit *texte* (chaîne de caractères) en (x, y).

5 Variables

Les identificateurs (noms de variables, procédures, etc.) peuvent contenir des lettres, des chiffres et les caractères « _ » et « $ ». Le premier caractère est obligatoirement une lettre. IDL ne fait pas de distinction entre les majuscules et les minuscules pour les identificateurs.

- i **=** 12 : Définit i comme entier et lui attribue la valeur 12 ($-32768 \leq i \leq 32767$).
- l **=** 12**L** : Définit l comme entier long.
- f **=** 12.5 ou f **=** 1.25**E**1 : Définit f comme réel simple précision et lui attribue la valeur 12,5.
- d **=** 1.25**D**1 : Définit d comme réel double précision et lui donne la valeur 12,5.
- s **=** *'abc'* ou s **=** *"abc"* : Définit s comme une chaîne de caractères et lui attribue la valeur *abc*.
- Entier impair, réel différent de 0, chaîne non vide : *Vrai*.
- Autres : *Faux*.

Les variables scalaires n'ont pas besoin d'être déclarées (contrairement aux tableaux). Lorsque le type d'une variable scalaire a besoin d'être connu avant usage, on peut lui attribuer une valeur provisoire du type approprié. Ainsi,

s **= ""**
read, s

permet de lire une chaîne de caractères s.

6 Tableaux

- i **= intarr(**n**,** p**)** : Tableau d'entiers $n \times p$. Le premier indice varie le plus vite. Le premier élément est i[0,0] (pas i[1,1]).
- l **= lonarr()** : Tableau d'entiers longs.
- f **= fltarr()** : Tableau de réels simple précision.
- d **= dblarr()** : Tableau de réels double précision.
- s **= strarr()** : Tableau de chaînes de caractères.
- i **= indgen(**n**)** : Crée le tableau d'entiers $i[0] = 0, i[1] = 1, \ldots, i[n-1] = n-1$.
- **findgen** (resp. **dindgen**) : Idem, mais le type du tableau est réel simple (resp. double) précision.
- **max(**A**,** i**)** : Maximum de A. Si i est présent, i est l'indice du maximum.
- **min(**A**)** : Idem pour le minimum.
- A**[**i**,** j**]** : Élément $A_{i,j}$.
- A**[**i**, *]** (resp. A**[*,** i**]**) : Tous les éléments de la i^{e} ligne (resp. colonne).
- A**[**i_1**:**i_2**,** j_1**:**j_2**]** : Section de tableau allant de l'indice i_1 à i_2 pour les lignes et de j_1 à j_2 pour les colonnes.
- A **= [1, 3, 5]** : Crée un tableau à une dimension (vecteur de taille 3) d'entiers avec $A[0] = 1$, $A[1] = 3$, $A[2] = 5$.
- A **= [[1, 2, 3], [4, 5, 6]]** : crée le tableau $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$.

  La plupart des opérations réalisables sur les scalaires peuvent être étendues aux tableaux. Il est préférable de travailler directement sur les tableaux dans leur ensemble plutôt que sur chacun de leurs éléments.

7 Opérateurs

- **not** : Non.
- **and** : Et.
- **or** : Ou inclusif.
- **xor** : Ou exclusif.
- **eq** : =.
- **ne** : ≠.
- **le** : ≤.
- **lt** : <.
- **ge** : ≥.
- **gt** : >.
- **^** : Puissance.
- *numérateur* **mod** *dénominateur* : Reste de la division entière.
- **/** : Division (entière si le numérateur et le dénominateur sont entiers).
- **+** : Addition ou concaténation pour des chaînes de caractères.
- **##** : Multiplication matricielle.
- $x < y$ (resp. $x > y$) : Retourne le minimum (resp. maximum) de x et y.

8 Quelques fonctions prédéfinies

- **float()** : Conversion en réel simple précision.
- **double()** : Conversion en réel double précision.
- **fix()** : Conversion en entier.
- **abs()** : Valeur absolue.
- **alog()** : Logarithme népérien.
- **alog10()** : Logarithme décimal.
- **exp()** : Exponentielle.
- **randomu(***générateur, n***)** : Retourne n (un seul si n est absent) nombres aléatoires distribués uniformément entre 0 et 1, où *générateur* est le nom d'une variable. Si cette variable n'est pas définie, le générateur est déterminé par l'horloge-système.
- **randomn()** : Idem pour des nombres aléatoires distribués selon une gaussienne de moyenne 0 et d'écart-type 1.
- **round()** : Arrondit à l'entier le plus proche.
- **sqrt()** : Racine carrée.

9 Divers

- **idl** : Pour lancer IDL.
- **exit** : Sortir d'IDL.
- **retall** : À taper avant de relancer le code si l'exécution s'est interrompue.
- **;** ... : Commentaire après le **;**.
- ... **$** : L'instruction continue à la ligne suivante.
- ... **&** ... : Pour séparer plusieurs instructions sur la même ligne.
- **?** : Aide en ligne.
- **help,** *variable* : Donne les caractéristiques de *variable* ou de toutes les variables si *variable* est absent.