Éléments de programmation des macros dans OpenOffice.org

Par

Andrew Pitonyak

Merci à ma femme Michelle Pitonyak pour m'avoir permis d'écrire ce livre et m'avoir encouragé. Merci à Laurent Godard et l'équipe de traduction française pour leurs bonnes idées et leur travail important. Robert Black Eagle, tu es infatigable et tu fais preuve d'une très grande perspicacité, sans mentionner ton excellent HowTo sur les documents maîtres. Kelvin Eldridge, merci pour m'avoir aidé à comprendre bon nombre de bugs. A Jean Hollis Weber et Solveig Haugland, merci pour vos réponses en privé quand j'avais des problèmes spécifiques sur l'utilisation des documents. A Sasa Kelecevic et Hermann Kienlein, merci de m'avoir fourni autant de documents utiles. A Andreas Bregas, merci pour la rapidité de tes réponses, explications et corrections. A Mathias Bauer, merci d'avoir pris le temps de m'expliquer et me fournir des exemples révélant les grands mystères des rouages internes. Enfin, je suis également redevable à la communauté open-source et aux personnes des listes de diffusion qui m'ont donné de précieuses informations.

Avertissement

Le contenu de ce document ne présente aucune garantie d'applicabilité, exactitude ou sécurité. L'utilisation de ce document et des macros incluses est faite à vos risques et périls. Vous reconnaissez que leur utilisation peut entraîner la destruction de vos bases de données, disques durs et la perte de vos données, ce dont je ne pourrais être tenu pour responsable.

Contact

Andrew Pitonyak • 4446 Mobile Drive #105 • Columbus, OH 43220 • USA Domicile:
andrew@pitonyak.org • Travail: andrew.pitonyak@qwest.com Téléphone personnel: 614-442-8615

L'auteur

J'ai deux licences de sciences, une en informatique, l'autre en mathématiques. J'ai également deux Masters de sciences, un en Mathématiques Industrielles appliquées, l'autre en informatique. J'ai séjourné à Oakland University (Rochester, Michigan), à Ohio State University (Columbus, Ohio) et à The Technical University Of Dresden (Dresden, Germany).

Public Documentation License

L'original de ce document est disponible sur http://www.pitonyak.org/AndrewMacro.sxw

Contribution	***Rédacteurs***	***Contact***	***Copyright***
Auteur Original	Andrew Pitonyak	andrew@pitonyak.org Andrew.Pitonyak@qwest.com	2002-2003
Traduction Française	Volontaires du projet French Native-lang	doc@fr.openoffice.org	2003
Macros	Hermann Kienlein	info@kienlein.com	2002-2003
Macros	Sasa Kelecvic	scat@teol.net	2002-2003

Note sur la traduction française

Le document d'Andrew est en constante évolution (autant dans le contenu que structurellement). Les volontaires du projet documentation ont décidé de ne pas reporter les modification apparues dans la version originale du document pendant la phase de traduction. Cette traduction s'appuie donc sur la version originale arrêtée au 14 novembre 2003.

Pour une version actualisée, le lecteur pourra se référer à la section du document original retraçant l'historique des modifications.

Nous vous souhaitons une bonne lecture ...

Les traducteurs

Table des matières

1Introduction

Quand j'ai voulu écrire ma première macro pour OpenOffice, je me suis noyé dans la complexité de l'API. Pour rendre la programmation de macros plus accessible, j'ai commencé une compilation de macros accomplissant des tâches élémentaires. Quand je voyais une requête pour une macro et que je ne savais pas comment faire, je prenais cela comme un défi, je l'écrivais et la documentais. Cette quête pour comprendre comment les macros fonctionnent dans OpenOffice est concrétisée par ce document.

La version la plus récente de ce document, mis à jour très fréquemment, peut être trouvée sur mon site :

http://www.pitonyak.org/AndrewMacro.sxw

La date de dernière modification se trouve sur la première page. La page principale de mon site Web indique également la date et l'heure de la dernière mise à jour. Ce document est basé sur un modèle également disponible sur mon site. Le modèle n'est pas requis pour lire le document, je ne le fournis que pour les plus curieux d'entre vous.

Dans ce document, OpenOffice.org est souvent abrégé en OOo. OOBasic est le nom du langage de macro disponible dans OpenOffice. OOBasic est très proche de Visual Basic, donc connaître ce langage sera d'un grand secours.

2Ressources disponibles

2.1Incluses dans OpenOffice

Ne sous-estimez pas la puissance des pages d'aide. On y trouve beaucoup d'informations sur la syntaxe des macros. Les pages d'aide sont classées en plusieurs sections. Après avoir affiché le sommaire de l'aide, déroulez la liste déroulante en haut à gauche, pour sélectionner “Aide sur OpenOffice.org Basic”.

Il est également instructif d'étudier et d'utiliser les macros fournies avec OpenOffice. On y trouve par exemple des macros donnant les propriétés et les noms des méthodes supportées par un objet. Quand j'ignorais quelles étaient les propriétés et méthodes d'un objet, j'ai utilisé ces macros pour trouver ce que je pouvais sur l'objet donné. Pour cela, ouvrez un document et choisissez le menu “Outils/Macros/Macro”. Dans la liste, cherchez un module intitulé “Tools”. Développez ce module, vous trouverez une entrée intitulée “Debug”. Ces macros implémentent la possibilité d'afficher des informations de débogage, sur les services, les attributs, etc... Regardez précisément les procédures *WritedbgInfo(document)* ou *printdbgInfo(sheet)* par exemple.

Pour utiliser la librairie de macros “Tools”, il faut tout d'abord la charger. Depuis l'EDI Basic (Interface de développement) ou depuis un document, choisissez le menu “OutilsMacros/Macro”, sélectionnez la librairie “Tools” et appuyez sur F5 ou cliquez sur “Exécuter”.

2.2Ressources en ligne

Il y a une grande richesse d'informations disponibles en ligne qui aide à décrypter la difficulté initiale de ce modèle de programmation. Voici quelques liens et références :

- http://fr.openoffice.org (lien principal) ;
- http://www.pitonyak.org/AndrewMacro.sxw (dernière copie à jour de ce document, en anglais)
- http://docs.sun.com/db/doc/817-1826-10 Sun a écrit un livre sur la programmation Basic. Il contient quelques erreurs (reprises dans le fichier d'aide), mais il reste excellent pour démarrer. Très bien écrit et présenté, en anglais ;
- http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html (beaucoups d'informations, en anglais) ;
- http://api.openoffice.org Ce site, en anglais, est quelque peu difficile à utiliser mais il est vraiment complet ;
- http://api.openoffice.org/basic/man/tutorial/tutorial.pdf (en anglais, excellent) ;
- http://udk.openoffice.org/common/man/tutorial/office_automation.html (en anglais) ;
- http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html API plus anciennes ;
- http://documentation.openoffice.org Télécharger le “how to” suivant, en anglais :

  http://documentation.openoffice.org/HOW_TO/various_topics/How_to_use_basic_macros.sxwhttp://fr.openoffice.org/Documentation/Index.html (excellent site en français). Télécharger le “how to” suivant, en français :

  http://fr.openoffice.org/Documentation/How-to/Basic/ht01_basic.sxw
- http://docs.sun.com/db/coll/999.2?q=star+office (documentation Sun originale).

Suivent d'autres sites contenant des exemples de code et généralement des informations intéressantes.

- http://kienlein.com/pages/oo.html (exemples, anglais, allemand) ;
- http://www.darwinwars.com/lunatic/bugs/oo_macros.html (Exemples) ;
- http://disemia.com/software/openoffice/ (exemples, en anglais) ;
- http://sourceforge.net/project/showfiles.php?group_id=43716 (Exemples et documentations, en anglais en en italien).

Quand je cherche une information spécifique, j'utilise habituellement le "Guide du développeur", le tutoriel et, ultimement, je lance une requête Google, comme par exemple "cursor OpenOffice". Quand je veux affiner la recherche, j'utilise “site: api.openoffice.org cursor” et je peux ainsi voir à quoi ressemble l'interface de cette fonctionnalité.

Si j'ai une idée du nom du package, j'essaye de deviner sa localisation sur le web. L'idée est que vous trouverez module-ix.html plutôt que index.html. Ainsi vous pourrez trouver

http://api.openoffice.org/docs/common/ref/com/sun/module-ix.html

mais pas

http://api.openoffice.org/docs/common/ref/com/sun/index.html

A partir de la page web d'un module, vous pouvez suivre les liens vers les sous-modules.

2.3Traductions

Lien	***Langue***
http://www.pitonyak.org/AndrewMacro.sxw	Anglais
http://fr.openoffice.org/Documentation/Guides/Indexguide.html	Français

3Pour débuter : les concepts

OOBasic est similaire à Visual Basic, aussi la connaissance de ce langage sera d'un grand secours. Le langage macro de OOo ressemble beaucoup à celui de Microsoft Office car ils sont tous deux basés sur le Basic. Ces deux langages permettent d'accéder au modèle objet respectif de la suite à laquelle ils appartiennent. Donc, en dehors de la syntaxe du langage elle-même, la comparaison devra s'arrêter là et on peut raisonnablement dire que les paradigmes des deux suites sont différents.

3.1Ma première Macro : Hello World

Ouvrir un nouveau document OOo. Aller dans le menu Outils/Macros/Macro. Ce menu ouvre la boite de dialogue des macros. Du côté gauche de cette boite de dialogue, rechercher le document ouvert préalablement. Il portera probablement le nom « Sans nom1 ». Cliquer sous ce nom où il est écrit "standard ». Cliquer sur le bouton "Nouveau" pour créer un nouveau module. Toujours utiliser le nom "Module1" proposé par défaut n'est probablement pas le bon choix. Quand vous auvez plusieurs documents ouverts contenant tous un "Module1" il risque d'être difficile de les différencier. Pour l'instant, intituler ce premier module "MonPremierModule". L'interface OOBasic (IDE) s'ouvre alors. Modifier le code pré-inscrit pour qu'il ressemble à ceci :

```
REM  *****  BASIC  *****

Sub Main
	Print "Hello World"
End Sub
```

appuyer sur le bouton "Exécuter" de la barre d'outils et... hop, vous venez d'exécuter votre première macro.

3.2Regrouper le code

OOBasic est basé sur des sous-routines et des fonctions. Elles sont implémentées avec les mots clés Sub et Function. Elles seront dénommées comme “Procédures” tout au long de cet ouvrage. Chaque procédure donne accès à un ensemble de fonctionnalités et peut appeler d'autres procédures (la récursivité n'est permise qu'à partir de la version 1.1). La différence entre une Sub et une Function est que cette dernière peut retourner une valeur et donc à ce titre est autorisée à figurer à droite de l'affectation d'une variable.

```
Sub HelloWorld
	MsgBox HelloWorldString()
End Sub
Function HelloWorldString() As String
	HelloWorldString  = "Hello World"
End Function
```

Une collection de procédures est contenue dans un module. Un document peut contenir plusieurs modules qui peuvent également exister indépendamment du document (module global). Une collection de modules est contenue dans une librairie (library).

3.3Créer et accéder aux objets dans OOBasic

Dans OpenOffice.org Basic, on accède au document ouvert existant ou à l'application par l'intermédiaire des deux variables ThisComponent et StarDesktop . Une fois que vous avez l'objet document, vous pouvez accéder à son interface. Voici un exemple simple :

```
.
Sub Example
```

```
Dim oText As Variant, oDoc  As Variant
oDoc  = ThisComponent          ' Récupère le document actif
oText = oDoc.Text              ' Accède au service TextDocument
```

Pour charger un document existant ou un nouveau document vierge, l'objet desktop possède la méthode loadComponentFromURL(). Les structures OOBasic peuvent être créées pendant leur déclaration comme ceci (*NdT : attention, la casse de PropertyValue est importante*) :

```
Dim args1(2) as new com.sun.star.beans.PropertyValue      'Array 0 to 2
```

Pour créer une instance d'un service, utiliser la méthode globale createUnoService().

```
Sub PerformDispatch(vObj, uno$)
    Dim vParser, vDisp
    Dim oUrl As New com.sun.star.util.URL
    oUrl.Complete = uno$

    vParser = createUnoService("com.sun.star.util.URLTransformer")
    vParser.parseStrict(oUrl)

    vDisp = vObj.queryDispatch(oUrl,"",0)
    If (Not IsNull(vDisp)) Then vDisp.dispatch(oUrl,noargs())
End Sub
```

Astuce	Le guide du développeur préconise d'utiliser le type Variant plutôt que le type Object. Voir la section Dois-je utiliser Object ou Variant pour plus de détail.
Astuce	Vous pourrez rencontrer ce genre de code : `createUnoService("com.sun.star.frame.Desktop")` Cette ligne crée une instance de OOo. La variable StarDesktop étant en principe déjà disponible, ne vous en souciez pas et utilisez cette variable.

3.4Services

Dans OOo, la plupart des fonctionnalités sont implémentées en tant que services. Conceptuellement, un service est défini par une interface. Vous n'avez pas besoin de savoir comment un service est implémenté, seulement comment l'appeler. Un très bon exemple de service simple est le "Searchable" service. décrit ici :

http://api.openoffice.org/docs/common/ref/com/sun/star/util/XSearchable.html

Ce lien indique que les méthodes suivantes sont supportées :

Méthode	***Description***
createSearchDescriptor	Crée un descripteur définissant comment chercher.
findAll	Cherche toutes les occurrences
findFirst	Cherche la première occurrence
findNext	Cherche l'occurrence suivante

Ce lien indique également que tout objet TextDocument supporte cette interface. Il ne donne aucune indication sur comment la recherche est effectuée mais il dit comment utiliser une recherche. Il donne également des informations sur les valeurs retournées. Voir la section Rechercher et remplacer

Si vous manipulez un objet et que vous voulez savoir s'il supporte un certain service, vous pouvez appeler la méthode SupportsService(NomDuService). Par exemple, si vous voulez savoir si l'objet vDoc est un document texte, vous pouvez utiliser le code suivant :

```
If vDoc.supportsService("com.sun.star.sheet.TextDocument") Then
```

Malgré le risque d'évoquer des sujets un peu trop tôt dans ce document, faites attention à la gestion d'erreur comme mentionné plus loin. Que se passe-t-il si vDoc est de type Null ? Si vDoc est une structure ou s'il n'implémente pas la méthode SupportsService() ? Pour vous assurer contre de telles erreurs, pensez à initialiser un gestionnaire d'erreurs en utilisant la syntaxe "On Error" .

Vous trouverez plus d'informations sur comment obtenir des renseignements complémentaires à l'adresse :

http://api.openoffice.org/docs/common/ref/com/sun/star/lang/XServiceInfo.html

Si vous avez besoin d'un service et que vous n'avez pas une instance disponible, vous pouvez utiliser la méthode globale createUnoService(). Vous en trouverez des exemples tout au long de ce document.

3.5Examiner un objet

3.5.1De quel type est il ?

Il est utile connaître le type d'un objet afin de pouvoir l'utiliser correctement. Voici une brève liste des méthodes d'inspection :

Méthode	***Description***
IsArray	Le paramètre est il un tableau ?
IsEmpty	Le paramètre est il un variant non initialisé ?
IsNull	Le paramètre ne contient-t-il aucune donnée ?
IsObject	Le paramètre est il un objet OLE ?
IsUnoStruct	Le paramètre est il une structure UNO ?
TypeName	Quel est le nom du type de l'argument ?

Le nom du type du paramètre va donner ces informations :

Type	***TypeName()***
Variant	“Empty” ou le nom de l'objet contenu
Object	“Object” même si il est null. Idem pour les structures
regular type	Un nom de type régulier comme “String” ou “Long”
array	Nom suivi de parenthèses “()”

La macro suivante illustre ces méthodes :

```
Sub TypeTest
 Dim oSimpleFileAccess
 Dim aProperty As New com.sun.star.beans.Property
 oSimpleFileAccess = CreateUnoService( "com.sun.star.ucb.SimpleFileAccess" )
 Dim v, o As Object, s As String, ss$, a(4) As String
 ss = "Empty Variant: " & GetSomeObjInfo(v) & chr(10) &_
   "Empty Object: " & GetSomeObjInfo(o) & chr(10) &_
   "Empty String: " & GetSomeObjInfo(s) & chr(10)
 v = 4
 ss = ss & "int Variant: " & GetSomeObjInfo(v) & chr(10)
 v = o
 ss = ss & "null obj Variant: " & GetSomeObjInfo(v) & chr(10) &_
   "struct: " & GetSomeObjInfo(aProperty) & chr(10) &_
   "service: " & GetSomeObjInfo(oSimpleFileAccess) & chr(10) &_
   "array: " & GetSomeObjInfo(a())
```

```
  MsgBox ss, 64, "Type Info"
End Sub

REM  Retourne des informations de base pour le paramètre.
REM  Cela retourne également la dimension d'un tableau.
Function GetSomeObjInfo(obj) As String
  Dim s As String
  s = "TypeName = " & TypeName(vObj) & CHR$(10) &_
    "VarType = " & VarType(vObj) & CHR$(10)
  If IsNull(vObj) Then
    s = s & "IsNull = True"
  ElseIf IsEmpty(vObj) Then
    s = s & "IsEmpty = True"
  Else
    If IsObject(vObj) Then
      On Local Error GoTo DebugNoSet
      s = s & "Implementation = " & NotSafeGetImplementationName(vObj) & CHR$(10)
      DebugNoSet:
      On Local Error Goto 0
      s = s & "IsObject = True" & CHR$(10)
    End If
    If IsUnoStruct(vObj) Then s = s & "IsUnoStruct = True" & CHR$(10)
    If IsDate(vObj) Then s = s & "IsDate = True" & CHR$(10)
    If IsNumeric(vObj) Then s = s & "IsNumeric = True" & CHR$(10)
    If IsArray(vObj) Then
      On Local Error Goto DebugBoundsError:
      Dim i%, sTemp$
      s = s & "IsArray = True" & CHR$(10) & "range = ("
      Do While (i% >= 0)
        i% = i% + 1
        sTemp$ = LBound(vObj, i%) & " To " & UBound(vObj, i%)
        If i% > 1 Then s = s & ", "
        s = s & sTemp$
      Loop
      DebugBoundsError:
      On Local Error Goto 0
      s = s & ")" & CHR$(10)
    End If
  End If
  adp_GetObjTypeInfo = s
End Function

REM  Cela crée un gestionnaire d'erreur pour gérer le problème
REM  et renvoie quelque-chose  quoiqu'il arrive !
Function SafeGetImplementationName(vObj) As String
  On Local Error GoTo ThisErrorHere:
  SafeGetImplementationName = NotSafeGetImplementationName(vObj)
  Exit Function
ThisErrorHere:
  On Local Error GoTo 0
  SafeGetImplementationName = "*** Unknown ***"
End Function
```

```
REM  Le problème est que si cette fonction est appelée et que le type vObj
REM  ne supporte PAS l'appel getImplementationName(), alors je reçois
REM  une erreur "Object variable not set" lors de la définition de la fonction.
Function NotSafeGetImplementationName(vObj) As String
  NotSafeGetImplementationName = vObj.getImplementationName()
End Function
```

3.5.2Que peut faire cet Objet ?

Vous avez recherché son type et vous savez que vous avez un objet. Si vous avez une structure, il vous faut découvrir de quel type elle est pour connaître ses propriétés. Je fais ça généralement en ligne en cherchant sur le site web dédié à l'API. Les objets UNO supportent en général le service ServiceInfo comme mentionné plus haut. La méthode getImplementationName() de l'objet retourne le nom complet de l'objet. A partir de là, je recherche dans le guide du développeur ou sur Google pour trouver plus d'informations. La méthode getSupportedServiceNames() retourne la liste de toutes les interfaces supportées par l'objet. Pour découvrir ce que peut faire un objet, vous pouvez appeler ces trois méthodes :

```
MsgBox  vObj.dbg_methods
MsgBox  vObj.dbg_supportedInterfaces
MsgBox  vObj.dbg_properties
```

3.6Comment procéder avec les retours de type UNO

Utiliser les informations de Xserviceinfo comme mentionné auparavant ?

4Exemples

4.1Débogage et vérification de macros

Il est peut être difficile de déterminer quelles méthodes et propriétés sont disponibles pour un objet. Les méthodes de cette section devraient vous y aider.

4.1.1Déterminer le type d'un document

Dans OOo, la plupart des fonctionnalités sont définies par des services. Pour déterminer le type de document, regardez si le service les supportent. La macro suivante utilise cette manière de faire. Je suppose que c'est plus sûr qu'appeler ThisComponent.getImplementationName().

```
'*************************************************************
'Auteur : Inclu dans OpenOffice.org
'
Function GetDocumentType(oDoc)
	On Local Error GoTo NODOCUMENTTYPE
	If oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
		GetDocumentType() = "scalc"
	ElseIf oDoc.SupportsService("com.sun.star.text.TextDocument") Then
		GetDocumentType() = "swriter"
	ElseIf oDoc.SupportsService("com.sun.star.drawing.DrawingDocument") Then
		GetDocumentType() = "sdraw"
	ElseIf oDoc.SupportsService("com.sun.star.formula.FormulaProperties") Then
		GetDocumentType() = "smath"
	End If
	NODOCUMENTTYPE:
	If Err <> 0 Then
		GetDocumentType = ""
		Resume GOON
		GOON:
	End If
End Function
```

Cette macro, écrite à partir d'une décrite plus haut par Alain Viret [Alain.Viret@bger.admin.ch], retourne le filtre d'export pdf. Notez qu'elle peut aussi contrôler à partir des documents Impress.

```
Function GetPDFFilter(oDoc)
	On Local Error GoTo NODOCUMENTTYPE
	If
oDoc.SupportsService("com.sun.star.presentation.PresentationDocument")
Then
		GetPDFFilter() = "impress_pdf_Export"
	ElseIf oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument")
Then
		GetPDFFilter() = "calc_pdf_Export"
	ElseIf  oDoc.SupportsService("com.sun.star.text.TextDocument") Then
		GetPDFFilter() = "writer_pdf_Export"
	ElseIf  oDoc.SupportsService("com.sun.star.drawing.DrawingDocument")
Then
		GetPDFFilter() = "draw_pdf_Export"
	ElseIf  oDoc.SupportsService("com.sun.star.formula.FormulaProperties")
Then
```

```
        GetPDFFilter() = "math_pdf_Export"
    End If
    NODOCUMENTTYPE:
    If Err <> 0 Then
        GetPDFFilter = ""
        Resume GOON
        GOON:
    End If
End Function
```

4.1.2Afficher les Méthodes et Propriétés d'un objet

Cette excellente procédure affiche les noms des méthodes et propriétés supportées par un objet. Si le second paramètre est une chaîne vide, les noms des méthodes sont affichés. D'autres valeurs provoque l'affichage des noms des propriétés. Comme cet affichage est fréquemment très long, la liste est découpée en morceaux.

```
'************************************************************************
'Une procédure pour afficher toutes les méthodes et propriétés d'un objet
'************************************************************************
'Author : Tony Bloomfield
'email :  tonyb.lx@btinternet.com
'Modification : hal@thresholddigital.com pour supporter les services et vérifier qu'oObj existe
Sub DisplayMethods(oObj as Object, SWhat as String)
    Dim sMethodLIst as String, sMsgBox as String
    Dim fs, ep as Integer
    Dim i as Integer
    Dim EOL as Boolean
    If IsNull(oObj) Then
        print "L'objet n'existe pas."
    Else
        If sWhat = "m" Then
            sMethodList = oObj.DBG_Methods
        ElseIf sWhat = "s" Then
            sMethodList = oObj.DBG_SupportedInterfaces
        ElseIf sWhat = "p" Then
            sMethodLIst = oObj.DBG_Properties
        End If
        fs = 1
        EOL = FALSE
        While fs <= Len(sMethodList)
            sMsgBox = ""
            For i = 0 to 15
                ep = InStr(fs, sMethodList, ";")
                If ep = 0 then
                    ep = Len(sMethodList)
                End If
                sMsgBox = sMsgBox & Mid$(sMethodList, fs, ep - fs) & chr$(13)
                fs = ep + 1
            Next i
            MsgBox sMsgBox
```

```
            Wend
        End If
End Sub
```

4.1.3Explorateur d'objets (Version Andrew)

Notez que, bien que ces méthodes fonctionnent pour moi, vous n'avez pas les boîtes de dialogue, donc ça ne marchera pas pour vous. Même si je les fournissais, c'est loin d'être terminé. J'espère pouvoir terminer cet explorateur d'objet si j'arrive à m'en sortir avec les boîtes de dialogue.

```
Option Explicit
Sub ExampleCreateDialog
    Dim oObj As Object
    oObj = ThisComponent

    Dim oDlgDesc As Object, oDlg As Object
    DialogLibraries.LoadLibrary("Standard")
    ' Récupère la description de la boîte de dialogue à partir de la bibliothèque
    oDlgDesc = DialogLibraries.Standard
    Dim oNames(), i%
    oNames = DialogLibraries.Standard.getElementNames()
'   For i = lBound(oNames()) To uBound(oNames())
'       MsgBox "How about " + oNames(i)
'   Next
'   msgBox(DialogLibraries.dbg_methods)
'   msgBox(DialogLibraries.dbg_properties)
'   msgBox(DialogLibraries.dbg_supportedInterfaces)
    oDlgDesc = DialogLibraries.Standard.ObjectViewer
    ' Instancie la boîte de dialogue
    oDlg = CreateUnoDialog( oDlgDesc )

    Dim oModel As Object, oListBox As Object
    dim iCount As Integer, iPos As Integer, s$, j%, sNew$
    oModel = oDlg.Model

    AddToListBox(oObj.DBG_Properties, oDlg.GetControl("PropertiesBox"), ";")
    AddToListBox(oObj.DBG_Methods, oDlg.GetControl("MethodsBox"), ";")
    AddToListBox(oObj.DBG_SupportedInterfaces, oDlg.GetControl("InterfacesBox"), chr(10))

    ' Affiche la boîte de dialogue
    oDlg.execute
End Sub

Sub AddToListBox(s$, oList As Object, sep$)
    Dim iCount%, j%, iPos%, sNew$
    iCount = oList.ItemCount
    iPos = InStr(1, s, ":") + 1
    iPos = FirstNonWhiteSpace(iPos, s)
    Do While iPos <= Len(s)
        j = InStr(iPos, s, sep)
        If j = 0 Then j = Len(s)
        iPos = FirstNonWhiteSpace(iPos, s)
        sNew = Mid$(s, iPos, j - iPos)
```

```
            oList.addItem(sNew, iCount)
            iPos = j + 1
            iCount = iCount + 1
        Loop
End Sub
Function FirstNonWhiteSpace(ByVal i%, s$) As Integer
        If i <= Len(s) Then
            Do While IsWhiteSpace(Asc(Mid$(s, i, 1)))
                i = i + 1
                If i > Len(s) Then
                    Exit Do
                End If
            Loop
        End If
        FirstNonWhiteSpace = i
End Function
```

4.1.4Afficher les propriétés d'un objet dans une feuille de calcul

Il suffit de faire tourner la macro pour le croire. Elle crée un nouveau classeur et remplit les feuilles avec les informations sur l'objet. J'ai beaucoup modifié cette macro car elle utilisait des variables globales qui interféraient avec mes propres variables locales. Cela m'a permis de l'utiliser dans mon code existant.

```
'Author : Hermann Kienlein
'email :   info@kienlein.com
'online :  http://www.kienlein.com/pages/oo.html
Option Explicit     ' Oblige toute variable à être déclarée avant utilisation
Sub Main
        MainObjectDisplay(ThisComponent)
End Sub

' -------------------------------------------------------
'Crée un nouveau classeur et de nouvelles feuilles de calcul dedans. Nomme les feuilles
'et insère les informations dedans
Sub MainObjectDisplay(oObject As Object)
        Dim oInfo As Object, oDeskNeu As Object, oNewDoc As Object
        Dim sNewUrl As String, nSheetsUsed As Long
        Dim sInterfaces As String
        Dim NoArgs()

        nSheetsUsed = 0
        oDeskNeu = createUnoService("com.sun.star.frame.Desktop")
        'sNewUrl = "staroffice.factory:scalc"        ' Cette ligne ne marchait pas
        sNewUrl = "private:factory/scalc"
        oNewDoc = oDeskNeu.loadComponentFromURL( sNewUrl,"_blank",0,NoArgs())

        ObjInfo(oNewDoc, nSheetsUsed, "ThisComponent", oObject)

        On Local Error GoTo AllDone
        sInterfaces = oObject.dbg_supportedinterfaces

        On Local Error GoTo NoController
        If InStr(sInterfaces, "com.sun.star.frame.XModel") <> 0 Then
            oInfo = oObject.getCurrentController()
```

```
            ObjInfo(oNewDoc, nSheetsUsed, "getCurrentController", oInfo)
        End If
        NoController:

        On Local Error GoTo NoDocInfo
        '??
'       If InStr(sInterfaces, "com.sun.star.document.XDocumentInfoSupplier") <> 0 Then
            oInfo = oObject.getDocumentInfo()
            ObjInfo(oNewDoc, nSheetsUsed, "getDocumentInfo", oInfo)
'       End If
        NoDocInfo:

        On Local Error GoTo NoSelection
        '??
'       If InStr(sInterfaces, "com.sun.star.frame.XSelectionSupplier") <> 0 Then
            oInfo = oObject.getCurrentSelection()
            ObjInfo(oNewDoc, nSheetsUsed, "getCurrentSelection", oInfo)
'       End If
        NoSelection:

        On Local Error GoTo NoLibraryContainer
        If InStr(sInterfaces, "com.sun.star.script.XStarBasicAccess") <> 0 Then
            oInfo = oObject.getLibraryContainer()
            ObjInfo(oNewDoc, nSheetsUsed, "getLibraryContainer", oInfo)
        End If
        NoLibraryContainer:

        On Local Error GoTo NoViewData
        If InStr(sInterfaces, "com.sun.star.document.XViewDataSupplier") <> 0 Then
            oInfo = oObject.getViewData()
            ObjInfo(oNewDoc, nSheetsUsed, "getViewData", oInfo)
        End If
        NoViewData:

        On Local Error GoTo NoEvents
        If InStr(sInterfaces, "com.sun.star.document.XEventsSupplier") <> 0 Then
            oInfo = oObject.getEvents()
            ObjInfo(oNewDoc, nSheetsUsed, "getEvents", oInfo)
        End If
        NoEvents:
        AllDone:
        On Local Error GoTo 0
End Sub

' ------------------------------------------------------
Sub ObjInfo(oDoc As Object, nSheetsUsed&, sSheetName$, obj As Object)
        Dim i2 As Integer, bProp As Boolean
        Dim nRow&, nCol&, oSheet As Object
        nRow = 0
        ncol = 0
        'on error goto err_ObjInfo

        If Not isNull(obj) Then
```

```
            GoToNextSheet(oDoc, nSheetsUsed, sSheetName)
            oSheet = oDoc.Sheets.getByIndex(nSheetsUsed-1)
            'oCell.String=CStr(obj.WindowServiceName)
            SetCell(nCol, nRow, oSheet, TypeName(obj), 1, 0)
            SetCell(nCol, nRow, oSheet, VarType(obj), -1, 2)
            i2 = InStr(obj.dbg_methods,"getPropertySetInfo")
            If i2 > 0 Then
                bProp = true
            Else
                bProp = false
            End If
            ListItems (nCol, nRow, obj.dbg_methods, ";", obj, true, oSheet)
            MoveRow(nCol, nRow, 2)
            If Not isNull (obj.dbg_properties) Then
                'NextSheet()
                ListItems (nCol, nRow, obj.dbg_properties, ";", obj, false, oSheet)
            End If
            If bProp Then
                GetProps(nCol, nRow, obj, oSheet)
                bProp = false
            End If
            MoveRow(nCol, nRow, 1)
            ListItems (nCol, nRow, obj.dbg_supportedinterfaces, chr$(10), obj, false, oSheet)
            'CleanDbg       'Actuellement aucune idée de ce à quoi cela sert
            'NextSheet()
        End If
exit_ObjInfo:
        Exit Sub
err_ObjInfo:
        'print err
        If err=423 Then
            obName = inputbox("L'objet n'a pas de nom."+" Quel nom doit-on donner à la nouvelle feuille ?")
            Resume Next
        Else
            msgbox error$, 16
            Resume exit_ObjInfo
        End If
End Sub

' ------------------------------------------------------
' ListItems – recherche de caractères de séparation dans la chaîne transmise
' et affichage ligne par ligne
' ------------------------------------------------------
Sub ListItems(nCol&, nRow&, itemstring$, sep$, oBj As Object, gt As Boolean, oSheet As Object)
        'dim oCurs As Object
        Dim frag As String, sleft As String, sPrf As String
        Dim act As Integer, ex As Integer, nextpos As Integer
        Dim nextchr As Integer, lstr As Integer, lfrag As Integer
        Dim ch As Integer
        lstr = Len(itemstring)
        act =1
        'Tout ce qu'il y a à gauche du premier deux points
        SetCell(nCol, nRow, oSheet, Left(itemstring,InStr(1,itemstring, ":")), 0, 0)
```

```
    'Si il n'y a pas deuxpoints alors c'est fini
    If InStr(1,itemstring,":") < 1 Then
        SetCell(nCol, nRow, oSheet, itemstring, 0, 0)
        Exit Sub
    End If
    act= act+InStr(1,itemstring,":")       ' on commence par le premier séparateur
    'act = act+1              ' puis on se positionne sur un caractère plus loin
    MoveRow(nCol, nRow, 1)
    While act<lstr

        nextpos=InStr(act,itemstring,sep)     'déterminer la position du premier ; après le deux points
        frag  = Mid(itemstring,act,nextpos-act)
        lfrag = Len(frag)
        act=act+lfrag+1
        frag = LTrim(frag)     'Aligner à gauche
        If frag > "" Then
            Do
                nextchr = Asc(Mid(frag,1,1)
                If nextchr = "10" Then
                    Mid(frag,1,1," ")
                    frag = LTrim(frag)     'Aligner à gauche
                Else
                    exit Do
                End If
            Loop
        End If
        MoveRow(nCol, nRow, 1)
        SetCell(nCol, nRow, oSheet, frag, 0, 0)
        If gt Then
            GtVal(nCol, nRow, frag,oBj, oSheet)
        End If
    Wend
    itemstring = ""
End Sub

' -------------------------------------------------------
' Subroutine pour relire les propriétés, l'objet doit supporter la méthode .PropertySetInfo
Sub GetProps(nCol&, nRow&, obj, oSheet As Object)
    Dim vVariant as Variant
    dim nVar As Integer
    dim mProperties as variant
    dim mProps1 as variant
    dim sItemDescription
    dim nCount As Integer
    dim iP As Integer, iP1 As Integer
    dim n$
    dim p
    dim tmp$
    dim j%
    dim vItem
    dim sString
    MoveRow(nCol, nRow, 2)
    mProperties = obj.PropertySetInfo.Properties
```

```
    nCount = UBound(mProperties)-LBound(mProperties) + 2
    SetCell(nCol, nRow, oSheet, "Properties With Values", 0, 1)
    SetCell(nCol, nRow, oSheet, "Name", 1, 0)
    SetCell(nCol, nRow, oSheet, "Value", -1, 1)
    For iP = LBound(mProperties) To UBound(mProperties)
        p = mProperties(iP)
        n$ = p.name
        vVariant = obj.getPropertyValue(n$)
        SetCell(nCol, nRow, oSheet, n$, 1, 0)
            nVar = VarType(vVariant)
        Select Case nVar
        Case 1  'isNull
            SetCell(nCol, nRow, oSheet, "NULL-VALUE", 0, 1)
        Case 9  'object
            If Not isNull (vVariant.dbg_properties) Then
                ListItems (nCol, nRow, vVariant.dbg_properties, ";", vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
            if not isNull (vVariant.dbg_supportedinterfaces) then
                ListItems (nCol, nRow, vVariant.dbg_supportedinterfaces, _
                    chr$(10), vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
            If Not isNull (vVariant.dbg_methods) Then
                ListItems (nCol, nRow, vVariant.dbg_methods, ";", _
                    vVariant, false, oSheet)
                MoveRow(nCol, nRow, 2)
            End If
        Case Else
            If IsArray(vVariant) Then
            tmp$ = ""
            For j% = LBound(vVariant) To UBound(vVariant)
                vItem = vVariant(j%)
                If IsNull(vItem) Then
                    sItemDescription = "NULL-Value"
                ElseIf IsObject(vItem) Then
                    If Not isNull(vItem.dbg_properties) Then
                        sItemDescription = CStr(vItem.dbg_properties)
                    End If
                Else
                    sItemDescription = cstr(vItem)
                End If
                tmp$ = tmp$ & sItemDescription
            Next j%
            ListItems(nCol, nRow, tmp$,";",vVariant,false, oSheet)
        Else
            SetCell(nCol, nRow, oSheet, cstr(vVariant), 0, 1)
        End If
        end select
        MoveRow(nCol, nRow, 1)
        MoveCol(nCol, nRow, -1)
    Next iP
End Sub
```

```
' ------------------------------------------------------
' GetValue – relire le contenu
' ------------------------------------------------------
Sub GtVal (nCol&, nRow&, sGVal As String, oBje As Object, oSheet As Object)
	dim is1 As Integer, iAr As Integer
	dim s1 As String, s2 As String, s3 As String
	dim aR1(10) as variant
	dim o1 As Object
	is1 =InStr(sGVal," ")   'Rechercher le premier espace
	s1 = Mid(sGval,1,is1)
	s2 = Mid(sGVal,1,is1," ")
	sGVal = LTrim(sGVal)

	is1 = InStr(sGVal," ")
	s2 = Mid(sGVal,1,is1)
	s1 = LTrim(s1)
	s1= RTrim(s1)
	s2 = LTrim(s2)
	s2 = RTrim(s2)
	Select Case s1
	Case "SbxSTRING"
		Select Case s2
		Case "getURL"
			s3 = oBje.getURL()
		Case "getLocation"
			s3 = oBje.getLocation()
		Case "getImplementationName"
			s3 = oBje.getImplementationName()
		Case "getUserFieldName"
			s3 = oBje.getUserFieldName(0)
		Case "getUserFieldValue"
			s3 = oBje.getUserFieldValue(0)
		Case Else
			s3 = s2
		End Select
		's3 = oBje.&s2
		'msgbox(CStr(oBje)&s2)
		MoveCol(nCol, nRow, 4)
		SetCell(nCol, nRow, oSheet, s3, -4, 0)
	Case "SbxBOOL"
		Select Case s2
		Case "hasControllersLocked"
			s3= CStr(oBje.hasControllersLocked())
		Case "isModified"
			s3= CStr(oBje.isModified())
		Case "AutoloadEnabled"
			s3= CStr(oBje.AutoloadEnabled())
		Case "hasElements"
			s3= CStr(oBje.hasElements())
		Case "IsEncrypted"
			s3= CStr(oBje.IsEncrypted())
		Case "isReadonly"
```

```
            s3= CStr(oBje.isReadonly())
        Case Else
            s3 = "   "
        End Select
        MoveCol(nCol, nRow, 4)
        SetCell(nCol, nRow, oSheet, s3, -4, 0)
    Case "SbxINTEGER"
        Select Case s2
        Case "getUserFieldCount"
            s3 = CStr(oBje.getUserFieldCount())
        Case "EditingCycles"
            s3 = CStr(oBje.EditingCycles())
        Case Else
            s3 = ""
        End Select
        MoveCol(nCol, nRow, 4)
        SetCell(nCol, nRow, oSheet, s3, -4, 0)
    Case "SbxLONG"
        Select Case s2
        Case "getCount"
            s3 = CStr(oBje.getCount())
        Case Else
            s3 = ""
        End Select
        MoveCol(nCol, nRow, 4)
        SetCell(nCol, nRow, oSheet, s3, -4, 0)
    Case "SbxOBJECT"
        Select Case s2
        Case "getElementType"
            s3 = CStr(VarType(oBje.getElementType()))
            MoveCol(nCol, nRow, 4)
            SetCell(nCol, nRow, oSheet, s3, -4, 0)
        Case "getText"
            o1 = oBje.getText()
            MoveCol(nCol, nRow, 4)
            SetCell(nCol, nRow, oSheet, o1.dbg_properties, 3, 0)
            SetCell(nCol, nRow, oSheet, o1.dbg_methods, -7, 0)
        Case Else
        End Select
    Case "SbxARRAY"
        Select Case s2
        Case "getImplementationId"
            aR1() = oBje.getImplementationId()
            MoveCol(nCol, nRow, 4)
            For iAr = LBound(oBje.getImplementationID()) To _
                UBound(oBje.getImplementationID())
                s3 = CStr(aR1(iAr))
                SetCell(nCol, nRow, oSheet, s3, 1, 0)
            next iAr
            MoveCol(nCol, nRow, -(4+1+UBound(oBje.getImplementationID())))
        Case "getArgs"
            '?? Pourquoi ceci est il décommenté pour afficher ?
            aR1() = oBje.getArgs()
```

```
        MoveCol(nCol, nRow, 4)
        For iAr = LBound(oBje.getArgs()) To UBound(oBje.getArgs())
            o1 = aR1(iAr)
            s3 = o1.dbg_properties
            'GetProps(aR1(iAr)
            'oCell.String = s3
            MoveCol(nCol, nRow, 1)
        Next iAr
        MoveCol(nCol, nRow, -(4+1+UBound(oBje.getArgs())))
    Case "getTypes"
        aR1() = oBje.getTypes()
        MoveCol(nCol, nRow, 4)
        '  For iAr = LBound(oBje.getTypes()) To Ubound(oBje.getTypes())
        For iAr = LBound(aR1()) To UBound(aR1())
            o1 = aR1(iAr)
            s3 = VarType(o1)
            SetCell(nCol, nRow, oSheet, s3, 1, 0)
        Next iAr
        MoveCol(nCol, nRow, -(4+1+UBound(oBje.getTypes())))
    Case "getElementNames"
        aR1() = oBje.getElementNames()
        MoveCol(nCol, nRow, 4)
        For iAr = LBound(oBje.getElementNames()) To _
            UBound(oBje.getElementNames())
            'o1 = aR1(iAr)
            's3 = VarType(o1)
            SetCell(nCol, nRow, oSheet, aR1(iAr), 1, 0)
        Next iAr
        MoveCol(nCol, nRow, -(4+1+UBound(oBje.getElementNames())))
    Case "getSupportedServiceNames"
        aR1() = oBje.getSupportedServiceNames()
        MoveCol(nCol, nRow, 4)
        For iAr = LBound(oBje.getSupportedServiceNames()) To _
            UBound(oBje.getSupportedServiceNames())
            'o1 = aR1(iAr)
            's3 = VarType(o1)
            SetCell(nCol, nRow, oSheet, aR1(iAr), 1, 0)
        Next iAr
        MoveCol(nCol, nRow, _
         -(4+1+UBound(oBje.getSupportedServiceNames())))
    Case "getPrinter"
        aR1() = oBje.getPrinter()
        MoveCol(nCol, nRow, 4)
        For iAr = LBound(oBje.getPrinter()) To UBound(oBje.getPrinter())
            o1 = aR1(iAr)
            s3 = CStr(VarType(aR1(iAr)))
            '?? On n'affiche jamais ceci
            MoveCol(nCol, nRow, 1)
        Next iAr
        MoveCol(nCol, nRow, -(4+1+UBound(oBje.getPrinter())))
    Case Else
        s3 = "  "
        MoveCol(nCol, nRow, 4)
```

```
                SetCell(nCol, nRow, oSheet, s3, -4, 0)
            End Select
        Case Else
            s3 = " "
            MoveCol(nCol, nRow, 4)
            SetCell(nCol, nRow, oSheet, s3, -4, 0)
        End Select
end sub

' ------------------------------------------------------
Sub SetCell(nCol&, nRow&, oSheet As Object, s$, colInc%, rowInc%)
    oSheet.getCellByPosition(nCol, nRow).String = s$
    If colInc <> 0 Then MoveCol(nCol, nRow, colInc%)
    If rowInc <> 0 Then MoveRow(nCol, nRow, rowInc)
End Sub

' ------------------------------------------------------
Sub MoveCol(nCol&, nRow&, i%)
    nCol = nCol + i
    If nCol < 0 Then
        nRow = nRow + 1
        nCol = 0
    End If
End Sub

' ------------------------------------------------------
Sub MoveRow(nCol&, nRow&, i%)
    nRow = nRow + i
    If nRow < 0 Then
        nRow = 0
    End If
End Sub

' ------------------------------------------------------
'Crée une nouvelle feuille si nécessaire avec son nom
Sub GoToNextSheet(oDoc As Object, nSheetsUsed&, sSheetName$, Optional nWhichSheet%)
    Dim oSheets As Object, oSheet As Object
    oSheets = oDoc.Sheets
    If isNumeric(nWhichSheet) Then
        oSheets.insertNewByName("Sheet"&CStr(oSheets.Count()+1), nWhichSheet)
        oSheet = oSheet.getByIndex(nWhichSheet)
    Else
        If nSheetsUsed > oSheets.Count() - 1 Then
            nSheetsUsed = oSheets.Count() - 1
            oSheets.insertNewByName("Sheet"&CStr(oSheets.Count()+1), _
                nSheetsUsed)
        End If
        oSheet = oSheets.getByIndex(nSheetsUsed)
        nSheetsUsed = nSheetsUsed + 1
    End If
    oSheet.Name = sSheetName
End Sub
```

4.2Dispatch: Utiliser Universal Network Objects (UNO)

Le guide du développeur et l'URL http://udk.openoffice.org sont de bonnes références dans votre quête de la compréhension de UNO. UNO est un modèle de composant offrant l'interopérabilité entre des langages, des modèles d'objet, des architectures matérielles différentes et des processus. Cet exemple utilise une commande UNO pour effectuer la commande "Annuler" (*NdT : le Undo*)

```
Sub UnoUndo
    PerformDispatch(ThisComponent.CurrentController.Frame, ".uno:Undo")
End Sub

Sub PerformDispatch(oObj As Object, uno$)
    Dim oParser As Object
    Dim oUrl As New com.sun.star.util.URL
    Dim oDisp As Object
    Rem Le service UNO est représenté  comme une URL
    oUrl.Complete = uno$

    Rem Analyse l'URL comme requis
    oParser = createUnoService("com.sun.star.util.URLTransformer")
    oParser.parseStrict(oUrl)

    Rem Regarde si la Fenêtre active supporte la commande UNO
    oDisp = oObj.queryDispatch(oUrl,"",0)
    If (Not IsNull(oDisp)) Then
        oDisp.dispatch(oUrl,noargs())
    Else
        MsgBox uno$ & " was not found"
    End If
End Sub
```

A partir de la version 1.1, ceci peut s'écrire

```
Sub Undo
    Dim oDisp as object
    oDisp = createUnoService("com.sun.star.frame.DispatchHelper")
    oDisp.executeDispatch(ThisComponent.CurrentController.Frame,".uno:Undo", "", 0, Array())
End Sub
```

La partie difficile est de connaître l'interface UNO et les paramètres à utiliser. Par exemple, les deux exemples suivants qui devraient marcher dans la version 1.1 :

```
Dim args1(2) as new com.sun.star.beans.PropertyValue
Dim args2(0) as new com.sun.star.beans.PropertyValue
args1(0).Name = "URL"
args1(0).Value = "my_file_name_.pdf"
args1(1).Name = "FilterName"
args1(1).Value = "writer_pdf_Export"
oDisp.executeDispatch(document, ".uno:ExportDirectToPDF", "", 0, args1())
```

```
' position to B3
args2(0).Name = "ToPoint"
args2(0).Value = "$B$3"
dispatcher.executeDispatch(document, ".uno:GoToCell", "", 0, args2())
dispatcher.executeDispatch(document, ".uno:Copy", "", 0, Array())
dispatcher.executeDispatch(document, ".uno:Paste", "", 0, Array())
```

4.2.1Le Dispatcher a changé dans la version 1.1

Hal Vaughan a demandé : “Est-ce juste moi, ou bien y a t-il une raison pour que le dispatcher ne fonctionne pas sur la plupart des fonctions de la version 1.0.3 ?” Et Mathias Bauer de répondre :

"Le Dispatcher utilise certaines fonctionnalités non présentes dans la branche OOo 1.0.x (Dispatcher Helper) comme par exemple le code pour exécuter un dispatch avec des paramètres."

4.2.2Les noms des Dispatch changeront-ils ?

Une autre question de Hal Vaughan et réponse de Mathias Bauer.

Les macros utilisant les noms des Dispatch et pas les nombres ne changeront pas entre les différentes versions de OOo.

4.2.3Utiliser le Dispatcher nécessite une interface utilisateur.

Encore une autre question de Hal Vaughan et réponse de Mathias Bauer.

Y a t-il une raison quelconque d'utiliser les appels d'API classique plutôt que le dispatcher avec le nom de la fonction ?

Les appels Dispatch ne marchent pas sans une interface utilisateur. Si OOo tourne en mode serveur réel (ce qui pourrait arriver avec la version 2.0) où les documents sont chargés et travaillés en script sans interface graphique, seules les macros utilisant les appels directs à l'API fonctionneront. Les appels directs à l'API sont plus puissants et donnent une meilleure vue des objets manipulés. A mon humble avis, le dispatcher ne devrait être utilisé que pour ces deux seules raisons :

1. l'enregistreur de macros
2. comme une échappatoire quand une API n'est pas disponible ou si elle est défectueuse.

5Exemples divers

5.1Afficher du texte dans la barre de statut

```
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
'Modifié par : Andrew Pitonyak
'Voici deux méthodes qui peuvent être utilisées
'pour obtenir la barre de statut
Function ProgressBar
  ProgressBar = ThisComponent.CurrentController.StatusIndicator
  'ou bien
  'ProgressBar =  StarDesktop.getCurrentComponent.StatusIndicator
   REM Le code suivant a été ajouté, mais n'a pas été testé !
   ProgressBar.reset
   ProgressBar.start("a label", MaxValue)
   aValue = 1
   ProgressBar.setValue(aValue)
End Function

REM affichage du texte dans la barre de statut
Sub StatusText(sInformation)
  Dim sSpaces As String
  Dim iLen,iRest As Integer
  'sSpaces=SPACE(270)
  iLen=Len(sInformation)
  iRest=270-iLen
  ProgressBar.start(sInformation+SPACE(iRest),0)
End Sub
```

D'après Christian Erpelding [erpelding@ce-data.de], la macro ci-dessus ne permet de changer la barre de statut qu'UNE SEULE FOIS, après cela, les changements sont oubliés. Utilisez setText plutôt que start, comme montré ci-dessous.

```
Sub StatusText(sInformation)
   Dim iLen,iRest As Integer
   iLen=Len(sInformation)
   iRest=350-iLen
   StatusBar.setText(sInformation+SPACE(iRest))
End Sub
```

5.2Afficher tous les modèles dans le document courant

Ce n'est pas aussi passionnant qu'il y paraît. Les modèles suivants existent pour un document texte : CharacterStyles, FrameStyles, NumberingStyles, PageStyles, et ParagraphStyles.

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub DisplayAllStyles
      Dim mFamilyNames As Variant
      Dim mStyleNames As Variant
      Dim sMsg As String
      Dim oFamilies As Object
```

```
    Dim oStyle As Object
    Dim oStyles As Object

    oFamilies = ThisComponent.StyleFamilies
    mFamilyNames = oFamilies.getElementNames()
    For n = LBound(mFamilyNames) To UBound(mFamilyNames)
        sMsg = ""
        oStyles = oFamilies.getByName(mFamilyNames(n))
        mStyleNames = oStyles.getElementNames()
        For i = LBound(mStyleNames) To UBound (mStyleNames)
            sMsg=sMsg + i + " : " + mStyleNames(i) + Chr(13)
            If ((i + 1) Mod 20 = 0) Then
                MsgBox sMsg,0,mFamilyNames(n)
                sMsg = ""
            End If
        Next i
        MsgBox sMsg,0,mFamilyNames(n)
    Next n
End Sub
```

5.3Itération au travers des documents ouverts

```
Sub Main
    Dim oDesktop As Object, oDocs As Object
    Dim oDoc As Object, oComponents As Object
    'Le hasMoreElements() échouera avec l'oDesktop,
    'Je ne sais pas pourquoi !
    'oDesktop = createUnoService("com.sun.star.frame.Desktop")
    oComponents = StarDesktop.getComponents()
    oDocs = oComponents.createEnumeration()
    Do While oDocs.hasMoreElements()
        oDoc = oDocs.nextElement()

    Loop
End Sub
```

NdT : Juste avant le loop, rajouter un "Print "test"" permet de mieux visualiser l'effet ;-). Il faut avoir plusieurs documents ouverts.

5.4Liste des Fontes et d'autres propriétés d'affichage

Astuce	En fabriquant une fonte, il est courant de générer des versions différentes pour les différents attributs de style comme le gras ou l'italique. Quand vous listez les fontes supportées par votre système, vous trouverez toutes ces variantes. Windows contient par exemple "Courier New", "Courier New Italic", "Courier New Bold", et "Courier New Bold Italic".

Voir également :

http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XToolkit.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XDevice.html

http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontDescriptor.html

```
'Auteur :      Paul Sobolik
'email :  psobolik@lycos.com
Sub ListFonts
    Dim oToolkit as Object
    oToolkit = CreateUnoService("com.sun.star.awt.Toolkit")

    Dim oDevice as Variant
    oDevice = oToolkit.createScreenCompatibleDevice(0, 0)

    Dim oFontDescriptors As Variant
    oFontDescriptors = oDevice.FontDescriptors

    Dim oFontDescriptor As Object

    Dim sFontList as String
    Dim iIndex as Integer, iStart As Integer, iTotal As Integer, iAdjust As Integer
    iTotal = UBound(oFontDescriptors) - LBound(oFontDescriptors) + 1
    iStart = 1
    iAdjust = iStart - LBound(oFontDescriptors)
    For iIndex = LBound(oFontDescriptors) To UBound(oFontDescriptors)
        oFontDescriptor = oFontDescriptors(iIndex)
        sFontList = sFontList & iIndex + iAdjust & ": " & oFontDescriptor.Name & " " &
oFontDescriptor.StyleName & Chr(10)
        If ((iIndex + iAdjust) Mod 20 = 0) Then
            MsgBox sFontList, 0, "Fonts " & iStart & " to " & iIndex + iAdjust & " of " & iTotal
            iStart = iIndex + iAdjust + 1
            sFontList = ""
        End If
    Next iIndex
    If sFontList <> "" Then MsgBox sFontList, 0, "Fonts " & iStart & " to " & iIndex & " of " & iTotal
End Sub
```

5.5Imprimer le document courant

Je me suis amusé avec ceci et j'ai pu imprimer. J'ai cessé de chercher comment imprimer du A4 sur mon imprimante au format lettre ! Je voulais paramétrer ceci par défaut mais j'ai décidé qu'il n'y avait pas lieu d'y consacrer trop de temps.

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
Sub PrintCurrentDocument
    Dim mPrintopts1(), x as Variant
    'Dimensionné à 0, si vous paramétrez autre chose, soyez certains de positionner ceci à  une
    'valeur plus élevée
    Dim mPrintopts2(0) As New com.sun.star.beans.PropertyValue
    Dim oDocument As Object, oPrinter As Object

    oDocument = ThisComponent

    '**********************************
    'Voulez vous choisir une imprimante particulière ?
    'Dim mPrinter(0) As New com.sun.star.beans.PropertyValue
```

```
    'mPrinter(0).Name="Name"
    'mPrinter(0).value="Other printer"
    'oDocument.Printer = mPrinter()

    '**********************************
    'Pour imprimer simplement les documents, faire ceci :
    'oDocument.Print(mPrintopts1())

    '**********************************
    'pour imprimer les  pages 1-3, 7, et 9
    'mPrintopts2(0).Name="Pages"
    'mPrintopts2(0).Value="1-3; 7; 9"
    'oDocument.Printer.PaperFormat=com.sun.star.view.PaperFormat.LETTER
    'DisplayMethods(oDocument, "propr")
    'DisplayMethods(oDocument, "")
    oPrinter = oDocument.getPrinter()
    MsgBox "printer is from " + LBound(oPrinter) + " to " + UBound(oPrinter)
    sMsg = ""
    For n = LBound(oPrinter) To UBound(oPrinter)
        sMsg = sMsg + oPrinter(n).Name + Chr(13)
    Next n
    MsgBox sMsg,0,"Print Settings"

    'DisplayMethods(oPrinter, "propr")
    'DisplayMethods(oPrinter, "")

    'mPrintopts2(0).Name="PaperFormat"
    'mPrintopts2(0).Value=com.sun.star.view.PaperFormat.LETTER
    'oDocument.Print(mPrintopts2())
End Sub
```

5.5.1Imprimer la page courante

```
dim aPrintOps(0) as new com.sun.star.beans.PropertyValue
oDoc = ThisComponent
oViewCursor = oDoc.CurrentController.getViewCursor()
aPrintOps(0).Name = "Pages"
aPrintOps(0).Value = trim(str(oViewCursor.getPage()))
oDoc.print(aPrintOps())
```

5.5.2Autres Arguments d'impression

Un autre paramètre à prendre en compte est le paramètre Wait avec une valeur de VRAI. Ceci permet l'impression synchrone et l'appel n'est pas renvoyé à la routine avant que l'opération d'impression ne soit terminée. Ceci permet d'éviter d'inclure un « listener » sur la fin de l'impression, si tant est que vous en ayez besoin.

5.5.3Définition de la Zone d'impression de Calc

Je ne sais pas comment déterminer la zone d'impression manuellement, mais je sais le faire par macro ! La solution réside dans l'utilisation d'une CellRangeAddress, qui comporte les propriétés suivantes.

Nom	Description
Sheet	index de la feuille
StartColumn	index de la colonne du bord gauche
StartRow	index de la ligne du bord supérieur
EndColumn	Index de la colonne du bord droite
EndRow	index de la ligne du bord inférieur

```
Dim oPrintArea(0) as New com.sun.star.table.CellRangeAddress
With oPrintArea(0)
  .Sheet = 0
  .StartColumn = 0
  .StartRow = 0
  .EndColumn = 14
  .EndRow = 91
End With
ThisComponent.Sheets.getByName("Sheet1").setPrintAreas(oPrintArea())
```

5.6Information de Configuration

5.6.1Changer la taille d'une Liste de Sélection

NdT : Cette macro permet de changer le nombre d'entrées dans la liste de fichiers récemment ouverts de OpenOffice.

Pour citer le site Web :

> Vous pouvez utiliser cette macro mais, tant que le problème ne sera pas résolu, vous devrez "tuer" votre instance d'OpenOffice.org. En effet, lors d'un arrêt normal, la modification effectuée par la macro n'est pas conservée.
> Il existe d'autres moyens, impliquant la manipulation directe des fichiers de configuration, mais ce n'est pas l'objet de notre discours dans ce document.

```
'Auteur : Unknown
'email :  http://ui.openoffice.org/howto/index.html
Option Explicit
Sub SetPickListNine
    ChangePickListSize( 9 )
End Sub

Sub ChangePickListSize( nSize As Integer )
    ' accède à l'objet de configuration
    Dim aConfigProvider As Object
    aConfigProvider = createUnoService( "com.sun.star.configuration.ConfigurationProvider" )

    ' Crée un objet pour le noeud d'historique
    Dim aHistorySettings As Object
    Dim aParams(0) As new com.sun.star.beans.PropertyValue
    aParams(0).Name = "nodepath"
    aParams(0).Value = "/org.openoffice.Office.Common/History"
    aHistorySettings = aConfigProvider.createInstanceWithArguments
( "com.sun.star.configuration.ConfigurationUpdateAccess", aParams() )

    ' Définit la taille de la liste
```

```
    aHistorySettings.replaceByName( "PickListSize", nSize )

    ' Valide les changements effectués
    aHistorySettings.commitChanges
End Sub
```

5.6.2Version de OOo

Malheureusement la fonction GetSolarVersion a tendance à ne pas changer, même lorsque la version de OOo change. La version 1.0.3.1 renvoie “641” et la 1.1RC3 renvoie 645, mais ceci ne suffit pas toujours pour donner assez de précisions. La macro suivante renvoie la version actuelle de OOo.

```
Function OOoVersion() As String
 'récupère la version de OOo en cours d'exécution
 'Auteur : Laurent Godard
 'e-mail : listes.godard@laposte.net
 '
 Dim aSettings, aConfigProvider
 Dim aParams2(0) As new com.sun.star.beans.PropertyValue
 Dim sProvider$, sAccess$
 sProvider = "com.sun.star.configuration.ConfigurationProvider"
 sAccess   = "com.sun.star.configuration.ConfigurationAccess"
 aConfigProvider = createUnoService(sProvider)
 aParams2(0).Name = "nodepath"
 aParams2(0).Value = "/org.openoffice.Setup/Product"
 aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

 OOOVersion=aSettings.getbyname("ooSetupVersion")
End Function
```

5.6.3OOo Locale

Cette macro retourne la langue dans laquelle OOo a été compilé. Cela permet de savoir donc dans quelle langue est l'interface utilisateur.

```
Function OOoLang() as string
 'récupère la version de OOo en cours d'exécution
 'Auteur : Laurent Godard
 'e-mail : listes.godard@laposte.net
 '
 Dim aSettings, aConfigProvider
 Dim aParams2(0) As new com.sun.star.beans.PropertyValue
 Dim sProvider$, sAccess$
 sProvider = "com.sun.star.configuration.ConfigurationProvider"
 sAccess   = "com.sun.star.configuration.ConfigurationAccess"
 aConfigProvider = createUnoService(sProvider)
 aParams2(0).Name = "nodepath"
 aParams2(0).Value = "/org.openoffice.Setup/L10N"
 aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

 Dim OOLangue as string
 OOLangue= aSettings.getbyname("ooLocale")    'en-US
 OOLang=lcase(Left(trim(OOLangue),2))         'en
End Function
```

5.7Ouvrir et fermer des documents (et l'application)

5.7.1Fermer OpenOffice et/ou des documents

Tous les documents OpenOffice et ses fenêtres (services) supportent l'interface XCloseable. Pour fermer ces objets, vous devrez appeler close(bForce As Boolean). Si bForce est faux, alors l'objet pourra refuser de se fermer. Si bForce est vrai, alors l'objet ne sera pas capable de refuser.

L'objet Bureau ne supporte pas l'interface XCloseable pour des raisons historiques. Cette méthode cause un événement queryTermination émis à tous les objets à l'écoute. Si aucun TerminationVetoException n'est positionné, un événement notifyTermination est émis et « vrai » est retourné. Sinon, un événement abortTermination est émis et « faux » est retourné. Pour citer Mathias Bauer, “la méthode terminate() a été utilisée pendant longtemps, bien avant que nous ne découvrions que ce n'est pas la bonne manière de manipuler les documents ou les fenêtres se fermant. Si cette méthode n'avait pas été là, nous aurions employé XCloseable pour le bureau également.”[Bauer001]

```
If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
    oDoc.close(true)
Else
    oDoc.dispose
End If
```

J'ai eu de Sasa Kelecevic [scat@teol.net] cette méthode, que je n'ai pas testée :

```
'------------ save_and_close -------------
'Utilisez l'une de ces deux méthodes
'oDocClose=StarDesktop.CurrentFrame.Close
'oDocClose=StarDesktop.ActiveFrame.Close
'------------ close_no_save ---------------------
'Utilisez l'une de ces deux méthodes
'oDocClose=ThisComponent.Dispose
'oDocClose=StarDesktop.ActiveFrame.Dispose
```

Pour fermer un document modifié sans sauvegarder, appelez la méthode setModified(False) avant de fermer le document. Dans OOo1.1, vous avez accès à une autre option : appeler la méthode Close (TRUE) du document ce qui fermera le document sans l'enregistrer, même si celui-ci a été modifié.

5.7.2Charger un document depuis une URL

Pour charger un document depuis une URL, utilisez la méthode LoadComponentFromURL() depuis le bureau. Ceci charge un composant dans un cadre nouveau ou existant.

Syntaxe :

loadComponentFromURL(
    string aURL,
    string aTargetFrameName,
    long nSearchFlags,
    sequence< com::sun::star::beans::PropertyValue > aArgs)

Valeur retournée :

com::sun::star::lang::XComponent

Paramètres :

aURL : URL du document à charger. Pour créer un nouveau document, utilisez "private:factory/scalc", "private:factory/swriter", etc.

aTargetFrameName : Nom du cadre qui contiendra le nouveau document. Si un cadre portant ce nom existe, il est utilisé, autrement il est créé. "_blank" crée un nouveau cadre, "_self" utilise le cadre courant, "_parent" utilise le cadre parent, et "_top" utilise le plus élevé des cadres du chemin courant dans l'arbre.

NsearchFlags : Utilisation des valeurs de FrameSearchFlag pour spécifier comment chercher le aTargetFrameName spécifié. Normalement, utilisez simplement 0.
http://api.openoffice.org/docs/common/ref/com/sun/star/frame/FrameSearchFlag.html

#	Name	Description
0	Auto	SELF+CHILDREN
1	PARENT	Inclut le cadre parent
2	SELF	Inclut le cadre de départ
4	CHILDREN	Inclut les cadres enfants du cadre de départ
8	CREATE	Le cadre sera créé si non trouvé
16	SIBLINGS	Inclut les autres cadres enfants du parent de cadre de départ
32	TASKS	Inclut tous les cadres de toutes les tâches dans la hiérarchie actuelle des cadres
23	ALL	Inclut tous les cadres non engagés dans d'autres tâches. 23 = 1+2+4+16 = PARENT + SELF + CHILDREN + SIBLINGS.
55	GLOBAL	Recherche dans toutes la hiérarchie de frames. 55 = 1+2+4+16+32 = PARENT + SELF + CHILDREN + SIBLINGS + TASKS.
63		GLOBAL + CREATE

Aargs : Indique le comportement spécifique de composant ou de filtre. "ReadOnly" avec une valeur booléenne indique si le document est ouvert en lecture seulement. "FilterName" indique le composant à créer ou le filtre à utiliser, par exemple : "scalc: Text - csv". Voir : http://api.openoffice.org/docs/common/ref/com/sun/star/document/MediaDescriptor.html

Exemple :

```
Rem Charge deux documents dans le même cadre
oDesk = createUnoService("com.sun.star.frame.Desktop")
Dim NoArgs()
Rem Le cadre "MyName" sera créé s'il n'existe pas car il inclut "CREATE"
oDoc1 = oDesk.LoadComponentFromUrl(sUrl_1, "MyName", 63, Noargs())
Rem Utilise un  cadre "MyName"  existant
oDoc2 = oDesk.LoadComponentFromUrl(sUrl_2, "MyName", 55, Noargs())
```

Astuce

Dans OOo1.1 le cadre implémente loadComponentFromURL, aussi vous pouvez utiliser :

```
oDoc = oDesk.LoadComponentFromUrl(sUrl_1, "_blank", 0, Noargs())
oFrame =    oDoc.CurrentController.Frame
oDoc = oFrame.LoadComponentFromUrl(sUrl_2, "", 2, Noargs())
```

Notez l'argument « drapeau » de la recherche et l'argument nom.

Attention

Dans OOo1.1 vous pouvez réutiliser un cadre seulement si vous connaissez son nom.

Exemple :

```
Sub insertDocumentAtCursor(sFileUrl As String, oText As Object, oDoc As Object)
    Dim oCur As Object
    Dim oProperties As Object
    oCur=oText.createTextCursorByRange(oDoc.getCurrentController().getViewCursor().getStart())
oCur.insertDocumentFromURL(sFileURL,oProperties)
End Sub
```

Exemple :

```
'------------ Ouvrir un nouveau document --------------------
```

```
'Dim NoArgs()
'oDocNew=StarDesktop.loadComponentFromURL("private:factory/swriter","_blank",0,NoArgs())'
---------- Ouvrir un document existant----------
'Dim NoArg()
'oDocOldFile=StarDesktop.loadComponentFromURL(sUrl,"_blank",0,NoArg())
```

Pour créer un nouveau document basé sur un modèle, utiliser le code suivant :

```
basic = createUnoService("com.sun.star.frame.Desktop")
args(0).Name = "AsTemplate"
args(0).Value = true
oDoc = basic.LoadComponentFromUrl("file:///C|/Templates%20Files/Special.stw","_blank",0,args())
```

Si vous désirez éditer le modèle, mettez « AsTemplate » à « False ».

5.7.3Comment activer des macros avec LoadComponentFromURL

Lorsqu'un document est chargé par une macro, les macros qui y sont contenues sont désactivées. C'est le réglage par défaut pour des questions de sécurité.. A partir de la version 1.1, vous pouvez activer des macros dès le chargement du document. Il faut mettre la propriété “MacroExecutionMode” à la valeur 2 ou 4 et cela devrait fonctionner. Ceci est basé sur un e-mail de la liste dev . Merci à Mikhail Voitenko <Mikhail.Voitenko@Sun.COM>

http://www.openoffice.org/servlets/ReadMsg?msgId=782516&listName=dev

Voici sa réponse sous forme condensée :

Il existe une propriété 'MediaDescriptor' qui s'appelle 'MacroExecutionMode', et qui utilise des valeurs provenant des constantes 'com.sun.star.document.MacroExecMode' . Si cette propriété n'est pas spécifiée, le comportement par défaut empêche l'exécution de la macro. Les valeurs constantes supportées sont données au lien suivant :
http://api.openoffice.org/source/browse/api/offapi/com/sun/star/document/MacroExecMode.idl?rev=1.5&content-type=text/x-cvsweb-markup

#	*Nom*	*Description*
0	NEVER_EXECUTE	Ne jamais exécuter
1	FROM_LIST	Exécuter les macros à partir d'une liste définie, la possibilité d'émettre un avertissement est donnée par la configuration générale.
2	ALWAYS_EXECUTE	Une macro sera toujours exécutée, la possibilité d'émettre un avertissement est donnée par la configuration générale
3	USE_CONFIG	Utiliser la configuration générale pour récupérer la configuration d'exécution de macro. Dans le cas où une confirmation de la part de l'utilisateur est nécessaire, une boîte de dialogue s'affiche.
4	ALWAYS_EXECUTE_NO_WARN	Une macro sera toujours exécutée sans avertissement.
5	USE_CONFIG_REJECT_CONFIRMATION	Utiliser la configuration générale pour récupérer la configuration d'exécution des macros. Cas où l'utilisateur a rejeté la demande de confirmation
6	USE_CONFIG_APPROVE_CONFIRMATION	Utiliser la configuration générale pour récupérer la configuration d'exécution des macros. Cas où l'utilisateur autorise la macro

Il existe quelques points sensibles qui méritent attention. Si vous chargez un document avec le paramètre "AsTemplate" (c-à-d en tant que modèle), celui-ci n'est pas ouvert, il est créé. Vous devez lier vos évènements à la commande “create document” (créer un document) plutôt que “open document” (ouvrir un document). Afin de couvrir les deux cas, vous pouvez lier la macro aux deux évènements.

```
Dim mFileProperties(1) As New com.sun.star.beans.PropertyValue
mFileProperties(0).Name="AsTemplate"
mFileProperties(0).Value=True
mFileProperties(1).Name="MacroExecutionMode"
mFileProperties(1).Value=4
```

Ceci devrait fonctionner pour des macros liées à l'évènement "OnNew" (Create Document), si vous chargez un modèle ou un document sxw (mais je ne l'ai pas essayé). Si vous utilisez "OnLoad" (Open Document), vous devez mettre "AsTemplate" à *False* (Faux) (ou bien utiliser un document sxw , parce que par défaut la valeur est mise à *False (Faux)*, alors que les modèles (stw) ont une valeur par défaut de *True (Vrai)*).

5.7.4Gestion d'erreur au chargement

Quand un document échoue au chargement, un message est affiché donnant des indications sur l'échec. Quand le document est chargé depuis C++, il est possible qu'aucune exception ne soit générée et vous ne serez pas informé de l'erreur.

Mathias Bauer a expliqué que l'interface XComponentLoader est incapable de générer une exception arbitraire et donc que le concept de “Interaction Handler” est utilisé. Quand un document est chargé par la méthode loadComponentFromURL, un “InteractionHandler” est passé dans le tableau d'argument. L'interface utilisateur donne un “InteractionHandler” qui converti les erreurs en interactions avec l'utilisateur comme afficher un message d'erreur ou demander un mot de passe (Voir le guide du développeur pour quelques exemples). Si aucun “InteractionHandler” n'est donné en argument, un « handler » par défaut est utilisé. Ce « Handler » par défaut intercepte toutes les erreurs et fait suivre les quelques unes qui pourraient être générées par loadComponentFromURL. Il est cependant impossible d'implémenter son propre Handler en OOBasic. Le guide du développeur donne des exemples dans d'autres langages.

5.7.5Exemple Pratique

Un publipostage crée un nouveau document pour chaque enregistrement de la base de données. La macro suivante récupère tous les documents Writer dans un seul répertoire et en fait un seul fichier contenant tous les documents du publipostage. J'ai modifié la macro d'origine de manière à ce que toutes les variables soient déclarées et ceci fonctionne même si le premier fichier trouvé n'est pas un document Writer.

```
'Auteur : Laurent Godard
'Modifié par : Andrew Pitonyak
Sub MergeDocumentsInDirectory()
'  On Error Resume Next
  Dim DestDirectory As String
  Dim FileName As String
  Dim SrcFile As String, DstFile As String
  Dim oDesktop, oDoc, oCursor, oText
  Dim argsInsert()
  Dim args()
  'Enlever les commentaires suivants afin de faire l'opération sous forme cachée
  'dim args(0) as new com.sun.star.beans.PropertyValue
  'args(0).name="Hidden"
  'args(0).value=true

  'Quel répertoire cible ?
  DestDirectory=Trim(GetFolderName())

  If DestDirectory = "" Then
    MsgBox "Aucun répertoire sélectionné, quittant l'opération",16,"Fusion des Documents"
```

```
  Exit Sub
 End If

 REM obliger l'insertion d'un anti-slash à la fin. Ceci fonctionne parce qu'on utilise la notation URL
 If Right(DestDirectory,1) <> "/" Then
  DestDirectory=DestDirectory & "/"
 End If

 oDeskTop=CreateUnoService("com.sun.star.frame.Desktop")

 REM Le code suivant lit le premier fichier !
 FileName=Dir(DestDirectory)
 DstFile = ConvertToURL(DestDirectory & "ResultatFusion.sxw")
 Do While FileName <> ""
  If lcase(right(FileName,3))="sxw" Then
    SrcFile = ConvertToURL(DestDirectory & FileName)
    If IsNull(oDoc) OR IsEmpty(oDoc) Then
      FileCopy( SrcFile, DstFile )
      oDoc=oDeskTop.Loadcomponentfromurl(DstFile, "_blank", 0, Args())
      oText = oDoc.getText
      oCursor = oText.createTextCursor()
    Else
      oCursor.gotoEnd(false)
      oCursor.BreakType = com.sun.star.style.BreakType.PAGE_BEFORE
      oCursor.insertDocumentFromUrl(SrcFile, argsInsert())
    End If
  End If
  FileName=dir()
 Loop

 If IsNull(oDoc) OR IsEmpty(oDoc) Then
  MsgBox "Aucun document fusionné!",16,"Fusion des Documents"
  Exit Sub
 End If

 'Enregistrement du document
 Dim args2()
 oDoc.StoreAsURL(DestDirectory & "ResultatFusion.sxw",args2())
 If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
  oDoc.close(true)
 Else
  oDoc.dispose
 End If

 'Rechargez le document !
 oDoc=oDeskTop.Loadcomponentfromurl(DstFile,"_blank",0,Args2())
End Sub
```

5.8Créer une table

Je n'ai rien fait avec ces macros de Kienlein ? ?

```
Sub InsertNextItem(what, oCursor, oTable)
    Dim oCelle As Object
    'nom de la plage de cellules sélectionnées par ce curseur
```

```
    sName = oCursor.getRangeName()
    ' Le nom de  cellule, qui sera quelque chose comme D3
    oCelle = oTable.getCellByName(sName)
    oCelle.String = what
    oCursor.goRight(1,FALSE)
End Sub

Function CreateTable() As Object
    oDocument = StarDesktop.ActiveComponent
    oTextTable = oDocument.createInstance("com.sun.star.text.TextTable")
    CreateTable = oTextTable
End Function
```

5.9Appeler un programme externe

Utilisez la commande shell.

5.10Nom de fichier externe avec espaces

Voir la section sur la notation URL ! En résumé, utilisez un %20 là où devrait se trouver un espace.

```
Sub ExampleShell
    Shell("file:///C|/Andy/My%20Documents/oo/tmp/h.bat",2)
    Shell("C:\Andy\My%20Documents\oo\tmp\h.bat",2)
End Sub
```

5.11 Lire et écrire un nombre dans un fichier

Cet exemple lit un texte d'un fichier texte. Ce nombre est converti en nombre et incrémenté. Le nombre est alors réécrit dans le fichier sous forme de chaîne de caractères.

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub Read_Write_Number_In_File
    DIM CountFileName As String, NumberString  As String
    DIM LongNumber As Long, iNum As Integer
    Dim oDocument As Object
    CountFileName = "C:\Andy\My Documents\oo\NUMBER.TXT"
    NumberString  = "00000000"
    LongNumber = 0
    'Si erreur locale, on va à NoFile
    If FileExists(CountFileName) Then
        ON ERROR GOTO NoFile
        iNum = FreeFile
        OPEN CountFileName for input as #iNum
        LINE INPUT #iNum ,NumberString
        CLOSE #iNum
        MsgBox("Lu " & NumberString, 64, "Lu")
NoFile:
        If Err <> 0 Then
            Msgbox("Impossible de lire " & CountFileName, 64, "Erreur")
            NumberString = "00000001"
        End If
```

```
        On Local Error Goto 0
    Else
        Msgbox(CountFileName & " n'existe pas", 64, "Attention!")
        NumberString  = "00000001"
    End If

    ON ERROR GOTO BadNumber
    LongNumber = Int(NumberString)
    LongNumber = LongNumber + 1
BadNumber:
    If Err <> 0 Then
        Msgbox(NumberString & " n'est pas un nombre", 64, "Erreur")
        LongNumber = 1
    End If
    On Local Error Goto 0
    NumberString=Trim(Str(LongNumber))
    While LEN(NumberString) < 8
        NumberString="0"&NumberString
    Wend
    MsgBox("Le nombre est (" & NumberString & ")", 64, "Information")
    iNum = FreeFile
    OPEN CountFileName for output as #iNum
    PRINT #iNum,NumberString
    CLOSE #iNum
End Sub
```

5.12Créer un style de format de nombre

Si vous voulez un format de nombre particulier, alors vous pouvez soit déjà l'avoir, soit le créer si vous ne l'avez pas. Pour de plus amples informations sur les formats valides voir le contenu de l'aide avec le mot clé « formats de nombre ; formats ». Ils peuvent être très complexes.

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function FindCreateNumberFormatStyle (_
    sFormat As String, Optional doc, Optional locale)
    Dim oDocument As Object
    Dim aLocale as new com.sun.star.lang.Locale
    Dim oFormats As Object
    oDocument = IIf(IsMissing(doc), ThisComponent, doc)
    oFormats = oDocument.getNumberFormats()
    'Si vous choisissez de chercher des types, vous aurez à utiliser
    'com.sun.star.util.NumberFormat.DATE
    'Je pourrais utiliser les valeurs de locales stockées à
    'http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt
    'http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html
    'J'utilise une locale NULL locale et je lui laisse employer ce qui convient
    'D'abord, vérifier si le format de nombre existe
    If ( Not IsMissing(locale)) Then
        aLocale = locale
    End If
    formatNum = oFormats.queryKey (sFormat, aLocale, TRUE)
    MsgBox "Le format numérique courant est" & formatNum
```

```
    ' Si le format n'existe pas, alors on l'ajoute
    If (formatNum = -1) Then
        formatNum = oFormats.addNew(sFormat, aLocale)
        If (formatNum = -1) Then formatNum = 0
        MsgBox "Le nouveau format numérique est " & formatNum
    End If
    FindCreateNumberFormatStyle = formatNum
End Function
```

5.13Retourner un tableau de Fibonnaci

Cette fonction renvoie un tableau de nombres de Fibonnaci.

```
'*************************************************************
' http://disemia.com/software/openoffice/macro_arrays.html
' Renvoie une suite de nombres de Fibonnaci
' On présume que count est supérieur ou égal à deux, afin de simplifier le code
Function Fibonnaci( Count As Integer )
    Dim result( 1 to Count, 1 ) As Double
    result( 1, 1 ) = 0
    result( 2, 1 ) = 1

    For i = 3 to Count
        result( i, 1 ) = result( i - 2, 1 ) + result( i - 1, 1 )
    Next i

    Fibonnaci = result()
End Function
```

5.14Insérer un texte à la position d'un signet

```
oDoc.getBookmarks().getByName("<yourBookmarkName>").getAnchor.setString(
"ce que vous souhaitez insérer")
```

5.15Sauvegarder et exporter un document

Sauvegarder un document est vraiment simple. La macro suivante sauvegardera un document, mais seulement s'il a été modifié, s'il n'est pas en lecture seule et qu'un emplacement de sauvegarde est paramétré.

```
If (oDoc.isModified) Then
    If (oDoc.hasLocation And (Not oDoc.isReadOnly)) Then
        oDoc.store()
    End If
End If
```

Si le document doit être sauvegardé ailleurs, alors vous devez paramétrer quelques propriétés pour indiquer où et comment le document doit être sauvegardé.

```
Dim mFileProperties(0) As New com.sun.star.beans.PropertyValue
Dim sUrl As String
sUrl = "file:///complete/path/To/New/document"
Rem Mettre la valeur à vrai (c'est-à-dire remplacer False par True) pour écraser le document.
mFileProperties(0).Name = "Overwrite"
mFileProperties(0).Value = False
```

```
oDoc.storeAsURL(sUrl, mFileProperties())
```

Le code montré jusque là n'exportera pas le document avec un format différent. Pour faire cela, un filtre d'export particulier doit être défini et toutes les propriétés requises doivent être paramétrées. Vous devrez connaître le nom du filtre d'exportation et l'extension du fichier. Il y a une liste de filtres d'import et d'export à :

http://www.openoffice.org/files/documents/25/111/filter_description.html

et il y a pas mal d'infos intéressantes à :

http://oooconv.free.fr/engine/OOOconv.php .

Une méthode séparée est requise pour les filtres graphiques et le reste. Pour exporter en utilisant un format non graphique, utilisez un formulaire semblable à ce qui suit :

```
Dim args2(1) as new com.sun.star.beans.PropertyValue
args2(0).Name = "InteractionHandler"
args2(0).Value = ""
args2(1).Name = "FilterName"
args2(1).Value = "MS Excel 97"
Rem Change le filtre d'export
oDoc.storeToURL("file:///c|/new_file.xls",args2())
Rem Utilise l'extension de fichier correcte
```

Notez que j'ai utilisé une extension de fichier correcte et que j'ai spécifié le bon filtre d'export. C'est un peu différent pour les documents graphiques. Premièrement, vous instancez un GraphicExportFilter et vous lui dites d'exporter une page à la fois.

```
oFilter=CreateUnoService("com.sun.star.drawing.GraphicExportFilter")
Dim args3(1) as new com.sun.star.beans.PropertyValue
For i=0 to oDoc.drawpages.getcount()-1
    oPage=oDoc.drawPages(i)
    oFilter.setSourceDocument(opage)
    args3(0).Name = "URL"
    nom=opage.name
    args3(0).Value =  "file:///c|/"&oPage.name&".JPG"
    args3(1).Name = "MediaType"
    args3(1).Value = "image/jpeg"
    oFilter.filter(args3())
Next
```

5.16Champs utilisateurs

J'ai passé un peu de temps sur les champs utilisateurs et même si je ne comprends pas réellement tout ce qu'il y a savoir, je peux au moins les utiliser ! La plupart des gens choisira d'utiliser des champs Maîtres (Master Fields) qui permettent de définir leurs propres noms et valeurs.

5.16.1Champs d'informations du document

Dans les propriétés du document, il existe 4 champs portant les noms “Info 1”, “Info 2”, “Info 3” et “Info 4”. Je ne les utilise pas mais comme ils existent et que vous pouvez y accéder, je les mentionne.

```
    ' Accède aux champs utilisateurs des propriétés du document
    vInfo = vDoc.getDocumentInfo()
    vVal = oData.ElementNames
    s = "===User Fields==="
    For i = 0 to vInfo.GetUserFieldCount() - 1
        sKey = vInfo.GetUserFieldName(i)
        sVal = vInfo.GetUserFieldValue(i)
```

```
        s = s & Chr$(13) & "(" & sKey & "," & sVal & ")"
    Next i
    '(Info 1,)
    '(Info 2,)
    '(Info 3,)
    '(Info 4,)
    MsgBox s, 0, "User Fields"
```

5.16.2Champs Texte

Je ne connais pas la raison d'être de ce type de champs. J'ai deux champs comme ça dans mon document de test mais ils n'ont aucune valeur associée.

```
    s = "===Text Fields==="
    Dim vEnum
    vEnum = vDoc.getTextFields().createEnumeration()
    If Not IsNull(vEnum) Then
        Do While vEnum.hasMoreElements()
            vVal = vEnum.nextElement()
            s = s & Chr(13) & "(" & vVal.TextFieldMaster.Name & ")"
            ' Je ne sais pas quoi faire avec ça  ???
        Loop
    End If
    MsgBox s, 0, "Text Fields"
```

5.16.3Champs Maîtres (Master Fields)

Les champs Maîtres sont sympas car vous pouvez y mettre vos propres valeurs, formules ou nombres. Cette section n'est qu'une brève investigation mais cela devrait suffire pour se lancer. Il y a 5 types de champs maîtres : « Illustration », « Table », « Text », « Drawing » et « User ». Les noms de ces champs commencent tous par “com.sun.star.text.FieldMaster.SetExpression.” suivi du type puis d'un autre point et enfin du nom du champ.

Voici un exemple simple :

```
vDoc = ThisComponent
sName = "Author Name"
If vDoc.getTextFieldMasters().hasByName("com.sun.star.text.FieldMaster.User." & sName) Then
    vField = vDoc.getTextFieldMasters().getByName("com.sun.star.text.FieldMaster.User." & sName)
    vField.Content = "Andrew Pitonyak"
    'vField.Value = 2.3      Rem Si vous préférez un nombre
Else
    vField = vDoc.createInstance("com.sun.star.text.FieldMaster.User")
    vField.Name = sName
    vField.Content = "Andrew Pitonyak"
    'vField.Value = 2.3      Rem Si vous préférez un nombre
End If
```

Cette macro affiche tous les champs maîtres d'un document :

```
Sub FieldExamples
    Dim vDoc, vInfo, vVal, vNames
    Dim i%, sKey$, sVal$, s$
    vDoc = ThisComponent
    Dim vTextFieldMaster
    Dim sUserType$
    sUserType = "com.sun.star.text.FieldMaster.User"
```

```
    vVal = vDoc.getTextFieldMasters()
    vNames = vVal.getElementNames()
    'vous pouvez avoir des noms tels que:
    'com.sun.star.text.FieldMaster.SetExpression.Illustration
    'com.sun.star.text.FieldMaster.SetExpression.Table
    'com.sun.star.text.FieldMaster.SetExpression.Text
    'com.sun.star.text.FieldMaster.SetExpression.Drawing
    'com.sun.star.text.FieldMaster.User
    s = "===Text Field Masters==="
    For i = LBound(vNames) to UBound(vNames)
        sKey = vNames(i)
        s = s & Chr$(13) & "(" & sKey
        vTextFieldMaster = vVal.getByName(sKey)
        If Not IsNull(vTextFieldMaster) Then
            s = s & "," & vTextFieldMaster.Name
            'Je n'ai pas vérifié si c'est le cas!
            If (Left$(sKey,Len(sUserType)) = sUserType) Then
                'Les types User ont egalement un type (double) et vous pouvez interroger
                ' pour savoir s'il s'agit d'expressions
                'http://api.openoffice.org/docs/common/ref/com/sun/star/text/FieldMaster/User.html
                s = s & "," & vTextFieldMaster.Content
            End If
        End If
        s = s & ")"
    Next i
    MsgBox s, 0, "Text Field Masters"
End Sub
```

Les routines suivantes ont été postées par Rodrigo V Nunes [rodrigo.nunes@net-linx.com] mais je ne les ai pas testées.

```
'=========================================================================
' CountDocVars – Routine comptant le nombre de variables document disponibles pour le document courant
' In -  DocVars: tableau des  variables document courantes (nom) présentes dans l'annonce
' DocVarValue: tableau des  variables document courantes (valeurs) présentes dans l'annonce
' Out – entier contenant le nombre total de variables doc trouvées
'=========================================================================
Function CountDocVars(DocVars , DocVarValue) As Integer
Dim VarCount As Integer
Dim Names as Variant

    VarCount = 0
    Names = thisComponent.getTextFieldMasters.getElementNames()
    for i%=LBound(Names) To UBound(Names)
        if (Left$(Names(i%),34) = "com.sun.star.text.FieldMaster.User") Then
            xMaster = oActiveDocument.getTextFieldMasters.getByName(Names(i%))
                DocVars(VarCount) = xMaster.Name
      DocVarValue(VarCount) = xMaster.Value
                VarCount = VarCount + 1   ' variable document créée par l'utilisateur
        End if
  Next i%
```

```
   CountDocVars = VarCount
End Function
'
'
'=============================================================================
' SetDocumentVariable -la routine utilisée pour créer/paramétrer la valeur d'une variable
'de document dans la liste de textfield utilisateur dans le document, sans
'insérer physiquement son contenu dans le texte de l'annonce
' In  - strVarName: chaîne contenant le nom de la variables à créer/paramétrer
'       aValue: chaîne avec la valeur de la variable document
' Out – drapeau booléen contenant le statut de l'opération: TRUE=OK,
' FALSE= la variable ne peut être créée ou modifiée
'=============================================================================
Function SetDocumentVariable(ByVal strVarName As String, ByVal aValue As String ) As Boolean
Dim bFound As Boolean

  On Error GoTo ErrorHandler
  oActiveDocument = thisComponent
  oTextmaster = oActiveDocument.getTextFieldMasters()
  sName = "com.sun.star.text.FieldMaster.User." + strVarName
  bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName)  ' teste si la variable existe.
   if bFound Then
            xMaster = oActiveDocument.getTextFieldMasters.getByName( sName )
            REM la valeur MEMBER est utilisée pour les valeurs décimales
            REM et la valeur CONTENT pour les chaînes de caractères
            'xMaster.value = aValue
            xMaster.Content = aValue
   Else       ' La variable document n'existe pas encore.
            sService = "com.sun.star.text.FieldMaster.User"
            xMaster = oActiveDocument.createInstance( sService )
            xMaster.Name = strVarName
            xMaster.Content = aValue
   End If
   SetDocumentVariable = True      'Succes
   Exit Function

ErrorHandler:
      SetDocumentVariable = False
End Function
'
'
'=============================================================================
' InsertDocumentVariable - routine insérant une variable de document dans la liste des textfields
'utilisateur du document 'et dans le texte d'annonce, à la position courante du curseur
' In  - strVarName: chaîne avec le nom r de la variable document à insérer
'       oTxtCursor: objet curseur courant avec la position où placer la variable doc
' Out - rien
'=============================================================================
Sub InsertDocumentVariable(strVarName As String, oTxtCursor As Object)

      oActiveDocument = thisComponent
      objField = thisComponent.createInstance("com.sun.star.text.TextField.User")
      sName = "com.sun.star.text.FieldMaster.User." + strVarName
```

```
      bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName)
      ' contrôle l'existence de la variable
   if bFound Then
            objFieldMaster =  oActiveDocument.getTextFieldMasters.getByName(sName)
            objField.attachTextFieldMaster(objFieldMaster)
            ' insère le champ texte
            oText = thisComponent.Text
            'oCursor = oText.createTextCursor()
            'oCursor.gotoEnd(false)
            oText.insertTextContent(oTxtCursor, objField, false)
      End If

End Sub

'=========================================================================
' DeleteDocumentVariable - routine éliminant une variable document de la liste des testfields 'utilisateur  du document'
' In  - strVarName: chaîne avec le nom de la variable document à éliminer
' Out - rien
'=========================================================================

Sub DeleteDocumentVariable(strVarName As String)

      oActiveDocument = thisComponent
      objField = oActiveDocument.createInstance("com.sun.star.text.TextField.User")
      sName = "com.sun.star.text.FieldMaster.User." + strVarName
      bFound = oActiveDocument.getTextFieldMasters.hasbyname(sName)
      ' contrôle l'existence de la variable
   if bFound Then
            objFieldMaster =  oActiveDocument.getTextFieldMasters.getByName(sName)
            objFieldMaster.Content = ""
            objFieldMaster.dispose()
      End If

End Sub
'
'=========================================================================
' SetUserVariable - fonction utilisée pour créer/paramétrer les variables utilisateurs dans le corps du document.
' Ces variables servent seulement à des usages/contrôles du système interne et ne sont PAS
' disponibles ou employées en Java (voir 'SetDocumentVariables' pour la création/le paramétrage
' de variables document partagées)
'
' In - strVarName: chaîne avec le nom de la variable document à créer/paramétrer. Si elle n'existe pas, elle sera créée
'      avalue: valeur variant avec le nouveau contenu de la variable définie dans strVarName
' Out - drapeau booléen contenant le statut de l'opération :
' TRUE=OK, FALSE=La variable, ne peut pas être créée ou modifiée
'=========================================================================

Function SetUserVariable(ByVal strVarName As String, ByVal avalue As Variant) As Boolean

Dim aVar As Variable
Dim index As Integer            'Index des noms de variables existants
Dim vCount As Integer
```

```
    On Error GoTo ErrorHandler
    'Vérifie que la variable document existe déja
    oDocumentInfo = thisComponent.Document.Info
    vCount = oDocumentInfo.getUserFieldCount()
    bFound = false
        For i% = 0 to (vCount - 1)
            If strVarName = oDocumentInfo.getUserFieldName(i%) Then
                bFound = true
                oDocumentInfo.setUserFieldValue(i%,avalue)
            End If
        Next i%
    If not bFound Then              'Variable document n'existant plus
        oDocumentInfo.setUserFieldName(i,strVarName)
        oDocumentInfo.setUserValue(i,avalue)
    End If
    ' teste si la valeur est supérieure au nombre de variables utilisateurs !

    SetUserVariable = True      'Success
    Exit Function

ErrorHandler:
    SetUserVariable = False
End Function
```

5.17Types définis par l'utilisateur

Bien qu'OOBasic permette syntaxiquement de déclarer vos propres types, il est incapable de les utiliser. Une Issue a été créée pour demander l'ajout de cette fonctionnalité (Bug ?)

http://www.openoffice.org/project/www/issues/show_bug.cgi?id=14465

```
Type PersonType
    Dim FirstName As String
    Dim LastName As String
End Type

Sub TypeExample
    Dim person As PersonType
    'On ne peut rien faire avec cette variable
End Sub
```

5.18Correcteur orthographique, césure et thésaurus

Faire une correction orthographique, appliquer les césures ou utiliser le thésaurus est très simple. Cependant, ces éléments retourneront une valeur Null s'ils ne sont pas configurés. Lors de mes tests initiaux, la césure a toujours retourné Null tant que je ne l'ai pas configurée dans les options.

```
Sub SpellCheckExample
    Dim s() As Variant
    Dim vReturn As Variant, i As Integer
    Dim emptyArgs(0) as new com.sun.star.beans.PropertyValue
    Dim aLocale As New com.sun.star.lang.Locale
    aLocale.Language = "en"
    aLocale.Country = "US"
```

```
    s = Array("hello", "anesthesiologist", "PNEUMONOULTRAMICROSCOPICSILICOVOLCANOCONIOSIS",
"Pitonyak", "misspell")

    '*********Exemple de correcteur orthographique !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XSpellChecker.html
    Dim vSpeller As Variant
    vSpeller = createUnoService("com.sun.star.linguistic2.SpellChecker")
    'Utilisez vReturn = vSpeller.spell(s, aLocale, emptyArgs()) si vous voulez des options!
    For i = LBound(s()) To UBound(s())
        vReturn = vSpeller.isValid(s(i), aLocale, emptyArgs())
        MsgBox "Spell check on " & s(i) & " returns " & vReturn
    Next

    '******Exemple de Césure !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XHyphenator.html
    Dim vHyphen As Variant
    vHyphen = createUnoService("com.sun.star.linguistic2.Hyphenator")
    For i = LBound(s()) To UBound(s())
        'vReturn = vHyphen.hyphenate(s(i), aLocale, 0, emptyArgs())
        vReturn = vHyphen.createPossibleHyphens(s(i), aLocale, emptyArgs())
        If IsNull(vReturn) Then
            'La césure est probalement désactivée
            MsgBox "Hyphenating " & s(i) & " returns null"
        Else
            MsgBox "Hyphenating " & s(i) & " returns " & vReturn.getPossibleHyphens()
        End If
    Next

    '****** Exemple Thesaurus !
    'http://api.openoffice.org/docs/common/ref/com/sun/star/linguistic2/XThesaurus.html
    Dim vThesaurus As Variant, j As Integer, k As Integer
    vThesaurus = createUnoService("com.sun.star.linguistic2.Thesaurus")
  s = Array("hello", "stamp", "cool")
    For i = LBound(s()) To UBound(s())
        vReturn = vThesaurus.queryMeanings(s(i), aLocale, emptyArgs())
        If UBound(vReturn) < 0 Then
            Print "Le Thesaurus n'a rien trouvé pour " & s(i)
        Else
            Dim sTemp As String
            sTemp = "Hyphenated " & s(i)
            For j = LBound(vReturn) To UBound(vReturn)
                sTemp = sTemp & Chr(13) & "Meaning = " & vReturn(j).getMeaning() & Chr(13)
                Dim vSyns As Variant
                vSyns = vReturn(j).querySynonyms()
                For k = LBound(vSyns) To UBound(vSyns)
                    sTemp = sTemp & vSyns(k) & " "
                Next
                sTemp = sTemp & Chr(13)
            Next
            MsgBox sTemp
        End If
    Next
End Sub
```

5.19Changer le curseur de la souris

La réponse rapide : ce n'est pas implémenté.

Une question sur le changement du curseur de la souris a induit une discussion intéressante que j'ai pris le temps de suivre mais pas de tester. J'ai édité les messages ci-dessous :

anindya@agere.com a demandé : Je voudrais que le curseur soit un sablier quand ma macro tourne. Qu'est ce qui ne va pas avec ce code ?

```
oDocument = oDeskTop.loadComponentFromURL(fileName,"_blank",0,mArg())
oCurrentController = oDocument.getCurrentController()
oFrame = oCurrentController.getFrame()
oWindow = oFrame.getContainerWindow()
oPointer = createUnoService("com.sun.star.awt.Pointer")
oPointer.SetType(com.sun.star.awt.SystemPointer.WAIT)
oWindow.setPointer(oPointer)
```

Mathias Bauer, que nous aimons tous, a répondu : On ne peut pas changer le curseur de la souris d'une fenêtre d'un document par l'API-UNO. VCL gère le pointeur de souris sur la fenêtre, pas la fenêtre mère. Toute fenêtre VCL peut avoir son curseur. Si vous voulez changer le pointeur de la souris de la fenêtre du document, vous devez avoir accès à son XwindowPeer et ce n'est pas disponible dans l'API. Un autre problème pourrait être qu'OOo change le pointeur en interne, écrasant vos changements.

Berend Cornelius donne la réponse finale : Votre routine marche très bien pour toute fenêtre fille de votre document. Le code suivant se réfère à un contrôle dans le document :

```
Sub Main
    GlobalScope.BasicLibraries.LoadLibrary("Tools")
    oController = Thiscomponent.getCurrentController()
    oControl =oController.getControl(ThisComponent.Drawpage().getbyIndex(0).getControl())
    SwitchMousePointer(oControl.getPeer(), False)
End Sub
```

Cette routine change le pointeur de la souris quand il est au-dessus du contrôle et revient à son état normal quand il le quitte. Vous voulez une fonction de « Wait » qui place le pointeur dans un état d'attente mais ce n'est pas possible avec l'API.

A mon avis, vous pouvez le changer mais pas pour tout :

```
oDoc.getCurrentController().getFrame().getContainerWindow().setPointer(...)
```

5.20Changer le fond de page

```
Sub Main
    ' Les familles de style
    oStyleFamilies= ThisComponent.getStyleFamilies()
    ' Les styles de page
    oPageStyles= oStyleFamilies.getByName("PageStyles")
    ' VOTRE style de page
    oMyPageStyle= oPageStyles.getByName("Standard")
    ' Votre fond
    with oMyPageStyle
        .BackGraphicUrl= _
            convertToUrl( <CheminVersVotreGraphique> )
        .BackGraphicLocation= _
            com.sun.star.style.GraphicLocation.AREA
    end with
End Sub
```

5.21Manipuler le presse-papier

?? Voir le guide du développeur page 331 !

Je ne connais pas la meilleure méthode pour accéder au presse-papier mais ce qui est présenté ici a marché, un jour, pour quelqu'un, quelque part...

Pour copier des données dans le presse-papier, il faut tout d'abord les sélectionner. L'interface optionnelle donne la possibilité de sélectionner des objets et d'accéder aux objets en cours de sélection. Cette section contient des exemples pour obtenir le texte sélectionné pour à la fois des documents Calc et Writer.

5.21.1Copier des cellules de Calc avec le presse-papier

Le premier exemple qui m'a été envoyé sélectionne des cellules dans une feuille de calcul et les colle dans une autre :

```
'Auteur : Ryan Nelson
'email : ryan@aurelius-mfg.com
'Cette macro copie une plage de cellules et la colle dans une feuille existante ou une nouvelle
Sub CopyPasteRange()
    'Inclure cette bibliotheque pour utiliser la commande DispatchSlot
    GlobalScope.BasicLibraries.LoadLibrary("Tools")

    Dim oSourceDoc As Object, oSourceSheet As Object, oSourceRange As Object
    Dim oTargetDoc As Object, oTargetSheet As Object, oTargetCell As Object
    Dim sUrl As String

    'Définit le document source/la feuille/la plage
    oSourceDoc=ThisComponent
    oSourceSheet= oSourceDoc.Sheets(0)
    oSourceRange = oSheet.getCellRangeByPosition(0,5,100,10000)

    'Sélectionne la plage source
    ThisComponent.CurrentController.Select(oSourceRange)

    'Copie la selection courante dans le presse-papier
    DispatchSlot(5711)

    oDesk = createUnoService("com.sun.star.frame.Desktop")
    'Définit l'URL du fichier cible ou ouvre une nouvelle feuille
    sUrl = "File:///C:/temp/testing2.sxc"

    'ouvre le fichier.
    Dim NoArg()
    oTargetDoc=oDesk.loadComponentFromURL(sUrl,"_blank",0,NoArg())
    oTargetSheet = oTargetDoc.Sheets(0)
    'Vous pouvez nettoyer la plage cible avant de coller si elle contient des données ou du formatage
    'Met le focus sur la cellule 0,0 avant de coller
    'On peut mettre le focus sur n'importe quelle cellule. Si on ne définit pas la position,
    ' le collage s'effectuera à partir de la dernière cellule active lors de la dernière fermeture du document
    oTargetCell = oTargetSheet.getCellByPosition(0,0)
    oTargetDoc.CurrentController.Select(oTargetCell)

    'Colle le presse-papier à la position courante
```

```
    DispatchSlot(5712)
End Sub
```

Cet exemple utilise la méthode DispatchSlot pour copier-coller du texte en utilisant le presse-papier. Une liste des « slots » supportés dans la version 1.0.3.1 peut être trouvée ici :

http://www.openoffice.org/files/documents/25/60/commands_103.html

Une autre possibilité est d'utiliser un dispatcher avec les arguments ".uno:Copy" et ".uno:Paste".

5.21.2Copier des cellules de Calc sans le presse-papier

Il est inutile d'utiliser le presse-papier quand on veut copier une plage de cellules dans la même feuille. Il est possible de copier, insérer, supprimer et déplacer une plage de cellule d'un endroit à l'autre dans la même feuille. Voir pour plus de détails :

http://api.openoffice.org/docs/common/ref/com/sun/star/sheet/XCellRangeMovement.html

Le code suivant a été posté sur la liste dev@api.openoffice.org

```
' Auteur :     Oliver Brinzing
' email :  OliverBrinzing@t-online.de
Sub CopySpreadsheetRange
    oSheet1 = ThisComponent.Sheets.getByIndex(0)    ' feuille no 1, originale
    oSheet2 = ThisComponent.Sheets.getByIndex(1)    ' feuille no 2

    oRangeOrg = oSheet1.getCellRangeByName("A1:C10").RangeAddress   ' copie la plage
    oRangeCpy = oSheet2.getCellRangeByName("A1:C10").RangeAddress   ' insère la plage

    oCellCpy = oSheet2.getCellByPosition(oRangeCpy.StartColumn,_
        oRangeCpy.StartRow).CellAddress ' Position d'insertion

    oSheet1.CopyRange(oCellCpy, oRangeOrg)                ' copie ...
End Sub
```

5.22Paramétrer la localisation

Dans OOo, les caractères sont localisés (*NdT : associés à une langue*). Dans le style « Macro Code » de ce document, j'ai mis la localisation sur inconnue et ainsi le texte n'est pas analysé par le correcteur orthographique. Pour dire à OOo qu'un mot est en français, on paramètre sa localisation à Français. On m'a demandé comment faire pour un document entier. Cela paraissait évident à première vue. Un curseur supporte les propriétés des caractères qui permettent de définir la localisation. J'ai donc créé un curseur, et paramétré la localisation. J'ai obtenu une erreur d'exécution. J'ai découvert que la propriété de localisation devait être de type Void (impliquant que l'on ne peut pas la paramétrer). Bien que mon essai suivant fonctionne pour mon document, vous devriez vérifier plus précisément, avec les tableaux entre autres.

```
Sub SetDocumentLocale
    Dim vCursor
    Dim aLocale As New com.sun.star.lang.Locale
    aLocale.Language = "fr"
    aLocale.Country = "FR"

    Rem Utilisation sous entendue d'un document writer
    Rem Récupere le composant Text du document
    Rem Crée le curseur sur le texte
    vCursor = ThisComponent.Text.createTextCursor()
    Rem Navigue au début du document
```

```
    Rem Navigue alors à la fin du document en sélectionnant tout le texte
    vCursor.GoToStart(False)
    Do While vCursor.gotoNextParagraph(True)
        vCursor.CharLocale = aLocale
        vCursor.goRight(0, False)
    Loop
    Msgbox "successfully francophonized"
End Sub
```

Il serait prudent d'utiliser “On Local Error Resume Next” mais je ne l'ai pas fait car cela aurait caché toute erreur durant mes tests préliminaires.

Vous devriez pouvoir définir la localisation pour le texte sélectionné ou un texte trouvé lors d'une recherche.

5.23AutoTexte

Je n'ai pas testé ce code, mais l'on m'a assuré qu'il fonctionnait. Vous ne pourrez pas utiliser ce code directement car il requiert une boite de dialogue non fournie mais la technique utilisée devrait être utile.

```
'Auteur : Marc Messeant
'email : marc.liste@free.fr
'Pour copier un autoTexte d'une groupe à l'autre
'ListBox1 : Le groupe initial
'ListBox2 : Le groupe destination
'ListBox3 : Elément du groupe initial à copier
'ListBox4 : Elément du groupe destination (pour information)

Dim ODialog as object
Dim oAutoText as object

' Cette procédure ouvre la boîte de dialogue et initialise la liste des groupes

Sub OuvrirAutoText
    Dim aTableau() as variant
    Dim i as integer
    Dim oListGroupDepart as object, oListGroupArrivee as object

    oDialog = LoadDialog("CG95","DialogAutoText")
    oListGroupDepart = oDialog.getControl("ListBox1")
    oListGroupArrivee = oDialog.getControl("ListBox2")
    oAutoText = createUnoService("com.sun.star.text.AutoTextContainer")
    aTableau = oAutoText.getElementNames()
    oListGroupDepart.removeItems(0,oListGroupDepart.getItemCount())
    oListGroupArrivee.removeItems(0,oListGroupArrivee.getItemCount())
    For i = LBound(aTableau()) To UBound(aTableau())
        oListGroupDepart.addItem(aTableau(i),i)
        oListGroupArrivee.addItem(aTableau(i),i)
    Next
    oDialog.Execute()
End Sub

'Ces 3 procédures sont appelées quand l'utilisateur sélectionne un groupe
' pour initialiser la liste des AutoTextes de chaque groupe
Sub ChargerList1()
```

```
    ChargerListeGroupe("ListBox1","ListBox3")
End Sub
Sub ChargerList2()
    ChargerListeGroupe("ListBox2","ListBox4")
End Sub

Sub ChargerListeGroupe(ListGroupe as string,ListElement as string)
    Dim oGroupe as object
    Dim oListGroupe as object
    Dim oListElement as object
    Dim aTableau() as variant
    Dim i as integer

    oListGroupe = oDialog.getControl(ListGroupe)
    oListElement = oDialog.getControl(ListElement)
    oGroupe = oAutoText.getByIndex(oListGroupe.getSelectedItemPos())
    aTableau = oGroupe.getTitles()
    oListElement.removeItems(0,oListElement.getItemCount())
    For i = LBound(aTableau()) To UBound(aTableau())
        oListElement.addItem(aTableau(i),i)
    Next
End Sub

'Transfère un élément d'un groupe vers un autre
Sub TransfererAutoText()
    Dim oGroupDepart as object,oGroupArrivee as object
    Dim oListGroupDepart as object, oListGroupArrivee as object
    Dim oListElement as object
    Dim oElement as object
    Dim aTableau() as string
    Dim i as integer

    oListGroupDepart = oDialog.getControl("ListBox1")
    oListGroupArrivee = oDialog.getControl("ListBox2")
    oListElement = oDialog.getControl("ListBox3")
    i =oListGroupArrivee.getSelectedItemPos()
    If oListGroupDepart.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un groupe de départ")
        Exit Sub
    End If
    If oListGroupArrivee.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un groupe d'arrivée")
        Exit Sub
    End If
    If oListElement.getSelectedItemPos() = -1 Then
        MsgBox ("Vous devez sélectionner un élément à copier")
        Exit Sub
    End If
    oGroupDepart = oAutoText.getByIndex(oListGroupDepart.getSelectedItemPos())
    oGroupArrivee = oAutoText.getByIndex(oListGroupArrivee.getSelectedItemPos())
    aTableau = oGroupDepart.getElementNames()
    oElement = oGroupDepart.getByIndex(oListElement.getSelectedItemPos())
    If oGroupArrivee.HasByName(aTableau(oListElement.getSelectedItemPos())) Then
```

```
            MsgBox ("Cet élément existe déja")
            Exit Sub
        End If
        oGroupArrivee.insertNewByName(aTableau(oListElement.getSelectedItemPos()),_
            oListElement.getSelectedItem(),oElement.Text)
        ChargerListeGroupe("ListBox2","ListBox4")
End Sub
```

5.24« Pieds » décimaux en fraction

On m'a demandé de convertir des macros Microsoft Office en macros OOo. J'ai décidé de les améliorer. Le premier jeu de macros prenait un nombre décimal de pieds et le convertissait en pieds et pouces. J'ai décidé d'écrire une routine plus générale, ignorant le code existant. Cela m'a également permis d'éviter quelques bugs dans le code existant. La manière la plus rapide que je connaisse pour réduire une fraction est de trouver le PGCD (*NdT : GCD en anglais*), le Plus Grand Commun Diviseur. La macro de fraction appelle la fonction GCD pour simplifier la fraction.

```
'e-mail : olivier.bietzer@free.fr
'Ceci utilise les algorithmes d'Euclide et c'est très rapide !
Function GCD(ByVal x As Long, ByVal y As Long) As Long
 Dim pgcd As Long, test As Long

 ' Nous devons avoir x >=y et des valeurs positives
 x=abs(x)
 y=abs(y)
 If (x < y) Then
  test = x : x = y : y = test
 End If
 If y = 0 Then Exit Function

 '  Euclide dit ....
 pgcd = y       ' par définition PGCD est le plus petit
 test = x MOD y  ' reste de la division
 Do While (test) ' Tant que le reste n'est pas 0
  pgcd = test    ' pgcd est le reste
  x = y          ' x,y et permutation courante de pgcd
  y = pgcd
  test = x MOD y ' nouveau test
 Loop
 GCD = pgcd  ' pgcd est le dernier reste différent de 0 ! Magique ...
End Function
```

La macro suivante détermine la fraction. Si x est négatif, alors le numérateur et la valeur retournée de x sont négatifs. Veuillez noter que le paramètre x est modifié.

```
'n: en sortie, contient le numérateur
'd: en sortie, contient le dénominateur
'x: Nombre à mettre en fraction en entrée, en sortie la partie entière
'max_d: Dénominateur maximum
Sub ToFraction(n&,  d&, x#, ByVal max_d As Long)
    Dim neg_multiply&, y#
    n = 0 : d = 1 : neg_multiply = 1 : y = Fix(x)
    If (x < 0) Then
        x = -x : neg_multiply = -1
    End If
    n = CLng((x - Fix(x)) * max_d)
```

```
    d = GCD(n, max_d)
    n = neg_multiply * n / d
    d = max_d / d
    x = y
End Sub
```

Pour tester cette routine, j'ai créé le code suivant :Sub FractionTest

```
    Dim x#, inc#, first#, last#, y#, z#, epsilon#
    Dim d&, n&, max_d&
    first = -10 : last  = 10 : inc = 0.001
    max_d = 128
    epsilon = 1.0 / CDbl(max_d)
    For x = first To last Step inc
        y = x
        ToFraction(n, d, y, max_d)
        z = y + CDbl(n) / CDbl(d)
        If abs(x-z) > epsilon Then Print "Conversion incorrecte " & x & " to " & z
    Next
End Sub
```

Bien que j'aie beaucoup ignoré le code initial, j'ai voulu conserver les formats d'entrée-sortie initiaux même s' ils ne sont pas adaptés.

```
Rem [-]feet'-inches n/d"
Rem Rien n'est retourné si c'est 0.
Function DecimalFeetToString64(ByVal x#) As String
    'J'utilise 64 car c'est ce qui était à l'origine
    DecimalFeetToString64 = DecimalFeetToString(x, 64)
End Function

Function DecimalFeetToString(ByVal x#, ByVal max_denominator&) As String
    Dim numerator&, denominator&
    Dim feet#, decInch#, s As String

    s = ""
    If (x < 0) Then
        s = "-"
        x = -x
    End If

    feet = Fix(x)         'Nombre entier de pieds
    x = (x - feet) * 12  'pouces
    ToFraction(numerator, denominator, x, max_denominator)
    Rem gère quelques traitements d'arrondis
    If (numerator = denominator AND numerator <> 0) Then
        numerator = 0
        x = x + 1
    End If

    If feet = 0 AND x = 0 AND numerator = 0Then
        s = s & "0'"
    Else
        If feet <> 0 Then
            s = s & feet & "'"
            If x <> 0 OR numerator <> 0 Then s = s &  "-"
```

```
            End If
            If x <> 0 Then
                s = s & x
                If numerator <> 0 Then s = s &  " "
            End If
            If numerator <> 0 Then s = s & numerator & "/" & denominator
            If x <> 0 OR numerator <> 0 Then s = s &  """"
        End If
        DecimalFeetToString = s
End Function

Function StringToDecimalFeet(s$) As Double
        Rem Le maximum de sortie devrait contenir
        Rem <pieds><'><-><pouces><espace><numérateur></><dénominateur><">
        Rem La première sortie doit être un nombre !
        Dim tokens(8) As String '0 to 8
        Dim i%, j%, num_tokens%, c%
        Dim feet#, inches#, n#, d#, leadingNeg#
        feet = 0 : inches = 0 : n = 0 : d = 1 : i = 1 : leadingNeg = 1.0
        s = Trim(s)    'Enlève les espaces superflus
        If (Len(s) > 0) Then
            If Left(s,1) = "-" Then
                leadingNeg = -1.0
                s = Mid(s, 2)
            End If
        End If

        num_tokens = 0 : i = 1
        Do While i <= Len(s)
            Select Case Mid(s, i, 1)
                Case "-", "0" To "9"
                    j = i
                    If Left(s, i, 1) = "-" Then j = j + 1
                    c = Asc(Mid(s, j, 1))
                    Do While (48 <= c And c <= 57)
                        j = j + 1
                        If j > Len(s) Then Exit Do
                        c = Asc(Mid(s, j, 1))
                    Loop
                    tokens(num_tokens) = Mid(s, i, j-i)
                    num_tokens = num_tokens + 1
                    i = j
                Case "'"
                    feet = CDbl(tokens(num_tokens-1))
                    tokens(num_tokens) = "'"
                    num_tokens = num_tokens + 1
                    i = i + 1
                    If (i <= Len(s)) Then
                        If Mid(s,i,1) = "-" Then i = i + 1
                    End If
                Case """", "/", " "
                    tokens(num_tokens) = Mid(s, i, 1)
                    i = i + 1
```

```
                Do While i < Len(s)
                    If Mid(s, i, 1) <> tokens(num_tokens) Then Exit Do
                    i = i + 1
                Loop
                If tokens(num_tokens) = "/" Then
                    n = CDbl(tokens(num_tokens-1))
                    num_tokens = num_tokens + 1
                ElseIf tokens(num_tokens) = " " Then
                    Inches = CDbl(tokens(num_tokens-1))
                ElseIf tokens(num_tokens) = """" Then
                    If num_tokens = 1 Then
                        Inches = CDbl(tokens(num_tokens-1))
                    ElseIf num_tokens > 1 Then
                        If tokens(num_tokens-2) = "/" Then
                            d = CDbl(tokens(num_tokens-1))
                        Else
                            Inches = CDbl(tokens(num_tokens-1))
                        End If
                    End If
                End If

            Case Else
                'Hmm, ceci est une erreur
                i = i + 1
                Print "In the else"
        End Select
    Loop

    If d = 0 Then d = 1
    StringToDecimalFeet = leadingNeg * (feet + (inches + n/d)/12)
End Function
```

5.25Envoyer un Email

OOo donne le moyen d'envoyer un document en pièce jointe par e-mail. OOo utilise un client existant plutôt que d'implémenter son propre protocole de mail. Sous Linux il devrait pouvoir utiliser les clients les plus courants comme Mozilla/Netscape, Evolution ou K-Mail. Sous Windows, OOo utilise MAPI donc tout client compatible devrait fonctionner. On va utiliser “com.sun.star.system.SimpleSystemMail”.

Cet exemple a été fourni par Laurent Godard. Comme lui, je n'ai pas réussi à mettre du texte dans le corps du mail généré, simplement envoyer une pièce jointe.

```
Sub SendSimpleMail()
    Dim vMailSystem, vMail, vMessage

    vMailSystem=createUnoService("com.sun.star.system.SimpleSystemMail")
    vMail=vMailSystem.querySimpleMailClient()
    'Pour en savoir plus sur les possibilités offertes
    'http://api.openoffice.org/docs/common/ref/com/sun/star/system/XSimpleMailMessage.html
    vMessage=vMail.createsimplemailmessage()
    vMessage.setrecipient("Andrew.Pitonyak@qwest.com")
    vMessage.setsubject("This is my test subject")

    'Les pieces jointes sont définies dans une séquence donc un tableau sous OOBasic
    'On aurait pu utiliser ConvertToURL() pour construite l'URL à partir du chemin système !
```

```
    Dim vAttach(0)
    vAttach(0) = "file:///c|/macro.txt"
    vMessage.setAttachement(vAttach())

    'DEFAULTS Lance le client mail par défaut du système
    'NO_USER_INTERFACE Pas d'interface utilisateur !
    'NO_LOGON_DIALOG Pas de boîte d'authentification – Génère une exception si une est requise
    vMail.sendSimpleMailMessage(vMessage, com.sun.star.system.SimpleMailClientFlags.NO_USER_INTERFACE)
End Sub
```

Ni le service SimpleSystemMail ni SimpleCommandMail ne sont capables de générer un contenu texte au mail. D'après Mathias Bauer, l'objectif de ces services est de pouvoir envoyer un document en tant que pièce jointe. Il est cependant possible d'utiliser une URL « mailto » avec un message dans le corps du mail mais qui ne contient pas de pièce jointe.

```
MyURL = createUnoStruct( "com.sun.star.util.URL" )
MyURL.Complete = "mailto:demo@someplace.com?subject=Test&Body=Text"
trans = createUnoService( "com.sun.star.util.URLTransformer" )
trans.parseStrict( MyURL )
disp = StarDesktop.queryDispatch( MyURL, "", 0 )
disp.dispatch( MyURL, noargs() )
```

Dans Ooo1.1, c'est encore plus facile :

```
dim noargs()
email_dispatch_url = "mailto:demo@someplace.com?subject=Test&Body=Text"
dispatcher = createUnoService( "com.sun.star.frame.DispatchHelper")
dispatcher.executeDispatch( StarDesktop,email_dispatch_url, "", 0, noargs() )
```

5.26Bibliothèques

Si vous désirez distribuer des bibliothèques et inclure des macros dans votre document pour les installer, un traitement spécial est requis. Sunil Menon a automatisé ce processus avec l'aide de Oliver Brinzing :

```
'Auteur :      Sunil Menon
'email :  sunil.menon@itb-india.com
Set service_name = "com.sun.star.script.ApplicationScriptLibraryContainer"
Set oLibLoad = objServiceManager.createInstance(service _name)
If Not oLibLoad Is Nothing Then
    On Error Resume Next
    If oLibLoad.isLibraryLoaded("mymacros") Then
        oLibLoad.removeLibrary ("mymacros")
    End If
    spath = "file:///D|/StarOfficeManual/mymacros"
    slib = "mymacros"
    Call oLibLoad.CreateLibraryLink(slib, spath, False)
    oLibLoad.loadLibrary ("mymacros")
    oLibLoad = Nothing
End If
```

Ce qui suit est un résumé de ce que Sun dit sur le sujet.

5.26.1Que signifie d'avoir une bibliothèque chargée ?

J'ai une application VB qui utilise l'interface de StarOffice. Plutôt que de coder des fonctionnalités comme « Chercher-Remplacer », « Imprimer » et « Extraction complexe de texte » en VB, j'utilise des

macros StarBasic. Pour distribuer ces macros aux utilisateurs, je crée des bibliothèques de macros (script.xlb, appmacro.xba). La bibliothèque est placée à l'endroit où mon application VB est installée. La bibliothèque doit être enregistrée et chargée avant d'être utilisable. Je peux alors appeler cette macro depuis Visual Basic en utilisant la commande Shell.

```
Shell "D:\StarOffice6.0\program\soffice.exe macro:///Standard.Module1.MAIN(""Hello Andrew"")", vbNormalFocus
```

5.26.2Pourquoi décharger la bibliothèque si elle est déjà chargée ?

Si je modifie les macros, elles doivent être de nouveau enregistrées avant que les changements ne soient pris en compte. Elles sont enregistrées à partir du répertoire de l'application et copiées ensuite par OOo.

5.26.3Quel est le rôle de l'appel à CreateLibraryLink ?

Cet appel crée un lien à une bibliothèque externe accessible en utilisant le gestionnaire de bibliothèques. Le format de l'URL dépend de l'implémentation. Le paramètre booléen est indicateur de lecture seule.

5.27Modifier la taille d'une Bitmap

Si vous chargez une image dans un document OOo, sa taille risque de ne pas convenir. Vance Lankhaar a attiré mon attention le premier sur ce problème. Sa première solution donnait une image de très petite taille :

```
'Auteur :      Vance Lankhaar
'email :  vlankhaar@linux.ca
Dim oDesktop As Object, oDocument As Object
Dim mNoArgs()
Dim sGraphicURL As String
Dim sGraphicService As String, sUrl As String
Dim oDrawPages As Object, oDrawPage As Object
Dim oGraphic As Object
sGraphicURL = "http://api.openoffice.org/branding/images/logonew.gif"
sGraphicService = "com.sun.star.drawing.GraphicObjectShape"
sUrl = "private:factory/simpress"
oDesktop = createUnoService("com.sun.star.frame.Desktop")
oDocument = oDesktop.loadComponentFromURL(sUrl,"_default",0,mNoArgs())
oDrawPages = oDocument.DrawPages
oDrawPage = oDrawPages.insertNewByIndex(1)
oGraphic = oDocument.createInstance(sGraphicService)
oGraphic.GraphicURL = sGraphicURL
oDrawPage.add(oGraphic)
```

La première solution donnée par Laurent Godard change la taille à la taille maximum possible :

```
'Taille maximum, perte du ratio de proportionnalité.
dim TheSize as new com.sun.star.awt.Size
dim TheBitmapSize as new com.sun.star.awt.Size
dim TheBitmap as object
dim xmult as double, ymult as double

TheBitmap=oGraphic.GraphicObjectFillBitmap
TheBitmapSize=TheBitmap.GetSize

xmult=TwipsPerPixelX/567*10*100 '567 twips = 1 cm *1*100 for 1/100th mm
ymult=TwipsPerPixelY/567*10*100
```

```
TheSize.width=TheBitmapSize.width*xmult
TheSize.height=TheBitmapSize.height*ymult

oGraphic.setsize(TheSize)
```

Vance Lankhaar en a déduit la solution finale maximisant la taille mais conservant le rapport de proportionnalité :

```
oBitmap = oGraphic.GraphicObjectFillBitmap
aBitmapSize = oBitMap.GetSize
iWidth  = aBitmapSize.Width
iHeight = aBitmapSize.Height

iPageWidth  = oDrawPage.Width
iPageHeight = oDrawPage.Height
dRatio = CDbl(iHeight) / CDbl(iWidth)
dPageRatio = CDbl(iPageHeight) / CDbl(iPageWidth)

REM C'est la dimension la plus grande de redimensionnement
If (dRatio < dPageRatio) Then
    aSize.Width = iPageWidth
    aSize.Height = CInt(CDbl(iPageWidth) * dRatio)
Else
    aSize.Width = CInt(CDbl(iPageHeight) / dRatio)
    aSize.Height = iPageHeight
End If

aPosition.X = (iPageWidth - aSize.Width)/2
aPosition.Y = (iPageHeight - aSize.Height)/2

oGraphic.SetSize(aSize)
oGraphic.SetPosition(aPosition)
```

5.27.1Insérer une Image, la Dimensionner, et la Positionner dans une Feuille de Calcul

David Woody [dwoody1@airmail.net] avait besoin d'insérer une image à une position et à une taille précises. Avec un peu d'aide et beaucoup de travail, il a réussi à élaborer la solution suivante :

Cette réponse m'a pris du temps, parce que je devais résoudre un autre problème lié à la détermination correcte des coordonnées X et Y. Le code suivant insère une image, la dimensionne, et la positionne à l'endroit voulu. J'ai dû ajouter la ligne suivante dans le code d'Andrew dans la section portant sur la spécification de la taille de l'image.

```
Dim aPosition as new com.sun.star.awt.Point
```

L'autre problème que j'ai eu, c'était de déterminer le rapport nécessaire entre aPosition.X et aPosition.Y afin de positionner correctement l'image. Sur mon ordinateur, la valeur de 2540 pour la coordonnée X ou Y était égale à un pouce à l'écran. Les valeurs ci-dessous positionneront l'image à un pouce du haut de la page et à un pouce depuis le bord gauche.

```
Sub InsertAndPositionGraphic
  REM Récupérer la feuille
  Dim vSheet
  vSheet = ThisComponent.Sheets(0)

  REM Insérer l'image
```

```
 Dim oDesktop As Object, oDocument As Object
 Dim mNoArgs()
 Dim sGraphicURL As String
 Dim sGraphicService As String, sUrl As String
 Dim oDrawPages As Object, oDrawPage As Object
 Dim oGraphic As Object
 sGraphicURL = "file:///usr/local/openoffice1.1RC/share/gallery/bullets/blkpearl.gif"
 sGraphicService = "com.sun.star.drawing.GraphicObjectShape"
 oDrawPage = vSheet.getDrawPage()
 oGraphic = ThisComponent.createInstance(sGraphicService)
 oGraphic.GraphicURL = sGraphicURL
 oDrawPage.add(oGraphic)

 REM Dimensionner l'image
 Dim TheSize as new com.sun.star.awt.Size
 TheSize.width=400
 TheSize.height=400
 oGraphic.setsize(TheSize)

 REM Positionner l'image
 Dim aPosition as new com.sun.star.awt.Point
 aPosition.X = 2540
 aPosition.Y = 2540
 oGraphic.setposition(aPosition)
End Sub
```

5.27.2Exporter une image à une Taille Prédéterminée

Cette macro est de Sven Jacobi [Sven.Jacobi@sun.com]

Il est possible de spécifier la résolution, mais elle n'est jamais parvenue au guide du développeur et j'en suis désolé. Cette possibilité existe à partir de la version 1.1 de OOo. Chaque filtre de conversion d'image supporte une séquence de propriétés appelée "FilterData" dans laquelle on peut préciser la taille en pixels avec les propriétés "PixelWidth" et "PixelHeight", la taille logique pouvant être définie(en 1/100mm) avec les propriétés "LogicalWidth" et "LogicalHeight". La macro suivante illustre cette possibilité.

```
Sub ExportCurrentPageOrSelection
 'création des propriétés dépendantes des filtres
 Dim aFilterData (4) as new com.sun.star.beans.PropertyValue
 aFilterData(0).Name  = "PixelWidth"    '
 aFilterData(0).Value = 1000
 aFilterData(1).Name  = "PixelHeight"
 aFilterData(1).Value = 1000
 aFilterData(2).Name  ="LogicalWidth"
 aFilterData(2).Value = 1000
 aFilterData(3).Name  ="LogicalHeight"
 aFilterData(3).Value = 1000
 aFilterData(4).Name  ="Quality"
 aFilterData(4).Value = 60
 Dim sFileUrl As String
 sFileUrl = "file:///d:/test2.jpg"

 xDoc = thiscomponent
 xView = xDoc.currentController
```

```
  xSelection = xView.selection
  If isEmpty( xSelection ) then
    xObj = xView.currentPage
  else
    xObj = xSelection
  End If
  Export( xObj, sFileUrl, aFilterData() )
End Sub

Sub Export( xObject, sFileUrl As String, aFilterData )
  xExporter = createUnoService( "com.sun.star.drawing.GraphicExportFilter" )
  xExporter.SetSourceDocument( xObject )

  Dim aArgs (2) as new com.sun.star.beans.PropertyValue
  Dim aURL as new com.sun.star.util.URL

  aURL.complete = sFileUrl
  aArgs(0).Name  = "MediaType"
  aArgs(0).Value = "image/jpeg"
  aArgs(1).Name  = "URL"
  aArgs(1).Value = aURL
  aArgs(2).Name  = "FilterData"
  aArgs(2).Value = aFilterData
  xExporter.filter( aArgs() )
End Sub
```

5.27.3Dessiner une Ligne dans un Document Calc

David Woody [dwoody1@airmail.net] a fourni la macro suivante :

Le code ci-dessous fonctionne pour moi. Notez toutefois que les variables 'TheSize' sont relatives à la variable 'aPosition', de telle sorte que si vous voulez x1 = 500 et x2 = 2000, alors TheSize.width = x2 - x1. La même chose s'applique pour la coordonnée Y.

```
Sub DrawLineInCalcDocument
  Dim xPage as object, xDoc as object, xShape as object
  Dim aPosition as new com.sun.star.awt.Point
  Dim TheSize as new com.sun.star.awt.Size

  xDoc = thiscomponent
  xPage = xDoc.DrawPages(0)
  xShape = xDoc.createInstance( "com.sun.star.drawing.LineShape" )
  xShape.LineColor = rgb( 255, 0, 0 )
  xShape.LineWidth = 100
  aPosition.X = 2500
  aPosition.Y = 2500
  xShape.setPosition(aPosition)
  TheSize.width = 2500
  TheSize.height=5000
  xShape.setSize(TheSize)
  xPage.add( xShape )
End Sub
```

5.28Extraction d'un Fichier Zippé

Laurent Godard [listes.godard@laposte.net] frappe encore et à nouveau avec cette solution. J'ai modifié

un peu son mail.

Bonjour à tous

Merci beaucoup de ton aide ! En combinant les conseils divers que vous m'avez tous donnés, j'ai enfin réussi à le faire fonctionner ! L'objectif est de gérer le contenu du flux de données rentrantes de la même manière que l'API de OOo, indépendamment de savoir ce qui est dedans !

Afin de résoudre mon problème, j'ai créé un flux sortant OutputStream et y ai écrit mon flux de données rentrantes, et c'est tout. Cela semble fonctionner (testé avec un fichier texte, mais devrait fonctionner pour les autres...). Comme promis, voici ma première tentative de la macro pour dézipper un fichier, dont le nom est connu, se trouvant dans un fichier ZIP. Il reste encore beaucoup de boulot à faire, mais cela peut permettre de faire avancer les choses.... Andrew, tu peux inclure ce code dans ta documentation sur les macros.

Merci à tous encore pour votre aide précieuse

Laurent Godard.

```
**************************
Sub UnzipAFile(ZipURL as string, SrcFileName as string, DestFile as string)
'Auteur : Laurent Godard
'E-maill : listes.godard@laposte.net

 Dim bExists as boolean

 ozip=createUnoService("com.sun.star.packages.Package")

 Dim args1(0)
 args1(0)=ConvertToURL(ZipURL)
 ozip.initialize(args1())

 'le fichier source existe-t-il ?
 bExists=ozip.HasByHierarchicalName(SrcFileName)
 if bnot bExists then exit sub

 'récupérer un flux de données
 ThePackageStream=ozip.GetByHierarchicalName(SrcFileName)

 'récupérer le flux de données rentrantes depuis SrcFileName
 MyInputStream=ThePackageStream.GetInputStream()

 'définir la sortie
 oFile = createUnoService("com.sun.star.ucb.SimpleFileAccess")
 oFile.WriteFile(ConvertToURL(DestFile),MyInputStream)

 'et voilà !!!

End Sub
```

5.28.1Un autre exemple sur un fichier zippé

Dan Juliano <daniel.juliano@rainhail.com> <djuliano@dmacc.edu> capitalise à partir de l'exemple de Laurent Godard. L'exemple suivant extrait tous les documents d'un fichier zippé.

```
' Instructions d'appel des subroutines suivantes
call unzipFileFromArchive("c:\test.zip", "test.txt", "c:\test.txt")
```

```
call unzipArchive("c:\test.zip", "c:\")

Sub unzipFileFromArchive( _
  strZipArchivePath As String, _
  strSourceFileName As String, _
  strDestinationFilePath As String)

  Dim blnExists          As Boolean
  Dim args(0)            As Variant
  Dim objZipService      As Variant
  Dim objPackageStream   As Variant
  Dim objOutputStream    As Variant
  Dim objInputStream     As Variant
  Dim i                  As Integer

  '=============================================================================
  ' Dézippe un fichier unique à partir d'une archive zippée. Vous devez connaître le nom exact du fichier
  ' présent dans l'archive pour que cette sub puisse le trouver
  '
  ' strZipArchivePath = chemin entier (répertoire et nom du fichier) identifiant l'archive .zip.
  ' strSourceFileName = nom du fichier à extraire de l'archive .zip.
  ' strDestinationFilePath = chemin entier (répertoire et nom du fichier) indiquant l'endroit
  '                         où le fichier doit être extrait.
  '=============================================================================

  ' Créer une instance pour le dézippeur
  objZipService = createUnoService("com.sun.star.packages.Package")
  args(0) = ConvertToURL(strZipArchivePath)
  objZipService.initialize(args())

  ' Le fichier source existe-t-il ?
  If Not objZipService.HasByHierarchicalName(strSourceFileName) Then Exit Sub

  ' Prendre l'information (stream) sur le fichier dans les données de l'archive
  objPackageStream = objZipService.GetByHierarchicalName(strSourceFileName)
  objInputStream = objPackageStream.GetInputStream()

  ' Définir l'extraction
  objOutputStream = createUnoService("com.sun.star.ucb.SimpleFileAccess")
  objOutputStream.WriteFile(ConvertToURL(strDestinationFilePath), objInputStream)
End Sub

Sub unzipArchive( _
  strZipArchivePath As String, _
  strDestinationFolder As String)

  Dim args(0)           As Variant
  Dim objZipService     As Variant
  Dim objPackageStream  As Variant
  Dim objOutputStream   As Variant
  Dim objInputStream    As Variant
  Dim arrayNames()      As Variant
  Dim strNames          As String
```

```
 Dim i               As Integer

 '=========================================================================================
 ' Dézippe une archive .zip entière dans un répertoire destination
 '
 ' strZipArchivePath = chemin entier (répertoire et nom du fichier) identifiant l'archive .zip.
  ' strDestinationFilePath = dossier destination (seulement le répertoire) dans lequel les fichiers de l'archive seront dézippés.
 '=========================================================================================

 ' Créer une instance pour le dézippeur
 objZipService = createUnoService("com.sun.star.packages.Package")
 args(0) = ConvertToURL(strZipArchivePath)
 objZipService.initialize(args())

 ' Récupérer les informations sur le contenu de l'archive
 objPackageStream = objZipService.GetByHierarchicalName("")

 ' Récupérer la liste de tous les fichiers zippés dans l'archive
 arrayNames = objPackageStream.getElementNames()

 ' Pour chaque fichier listé dans arrayNames, extraire/dézipper le fichier
' depuis l'archive vers le répertoire de destination
 For i = LBound(arrayNames) To UBound(arrayNames)
  strNames = strNames & arrayNames(i) & Chr(13)

  ' Lire et extraire un seul fichier à la fois pour le système de fichiers
  objInputStream = objZipService.GetByHierarchicalName(arrayNames(i)).GetInputStream()
  objOutputStream = createUnoService("com.sun.star.ucb.SimpleFileAccess")
  objOutputStream.WriteFile(ConvertToURL(strDestinationFolder & arrayNames(i)),_
    objInputStream)
 Next
 MsgBox strNames
End Sub
```

5.28.2Zipper des répertoires entiers

Laurent Godard fournit aussi cet exemple. Cette macro zippe le contenu d'un répertoire en incluant les sous-répertoires.

```
'-------------------------------------------------------------------
sub ExempleAppel
 call ZipUnRepertoire("C:\MesFichiers\Ooo\Rep","C:\resultat.zip")
end sub
'-------------------------------------------------------------------
sub ZipUnRepertoire(source as string, cible as string)
'Auteurr : Laurent Godard
'e-mail : listes.godard@laposte.net
 dim retour() as string

 'création de l'instance du fichier Zip
 LeFichierZip=createUnoService("com.sun.star.packages.Package")
 dim args(0)
 args(0)=ConvertToURL(cible)
```

```
 LeFichierZip.initialize(args())

 'création de la structure des répertoires dans le zip
 call Recursedirectory(source, retour)

 dim argsDir(0)
 argsDir(0)=true

 'Le premier élément le répertoire contenant --> on ne le traite pas dans la boucle
 'Pourra être une option à terme
 Repbase=retour(1)

 For i=2 To UBound(retour)
  chaine=mid(retour(i),len(repbase)+2)
  decoupe=split(mid(retour(i),len(repbase)+1),getPathSeparator)
  repZip=decoupe(ubound(decoupe))
  azipper=LeFichierZip.createInstanceWithArguments(argsDir())

  If len(chaine)<>len(repZip) then
    RepPere=left(chaine,len(chaine)-len(repZip)-1)
    RepPere=RemplaceChaine(reppere, getpathseparator, "/", false)
  Else
    RepPere=""
  Endif

  RepPereZip=LeFichierZip.getByHierarchicalName(RepPere)
  RepPereZip.insertbyname(repzip, azipper)
 Next i

 'insertion des fichiers dans les bons répertoires
 dim args2(0)
 args2(0)=false
 oUcb = createUnoService("com.sun.star.ucb.SimpleFileAccess")

 for i=1 to ubound(retour)
  chaine=mid(retour(i),len(repbase)+2)
  repzip=remplacechaine(chaine, getpathseparator, "/", false)
  fichier=dir(retour(i)+getPathSeparator,0)
  While fichier<>""
    azipper=LeFichierZip.createInstanceWithArguments(args2())
    oFile = oUcb.OpenFileRead(ConvertToURL(retour(i)+"/"+fichier))
    azipper.SetInputStream(ofile)
    RepPere=LeFichierZip.getByHierarchicalName(repZip)
    RepPere.insertbyname(fichier, azipper)
    fichier=dir()
  Wend
 next i

 'Valide les changements
 LeFichierZip.commitChanges()
 msgbox "C'est fini"
End Sub
'-------------------------------------------------------------------
```

```
sub RecurseDirectory(chemin, reponse as variant)
  'Auteur: Laurent Godard
  'e-mail : listes.godard@laposte.net
  'reponse est un tableau contenant la liste de tous les sous répertoires de chemin
  redim preserve reponse(1 to 1)
  compte=1
  reponse(1)=chemin
  repbase=1
  rep=dir(convertTourl(chemin+"/"),16)

  while rep<>""
    if rep<>"." and rep<>".." then
      compte=compte+1
      redim preserve reponse(1 to compte)
      reponse(compte)=convertfromurl(reponse(RepBase)+"/"+rep)
    endif
    rep=dir()

    while rep="" and repbase<compte
      repbase=repbase+1
      rep=dir(convertToURL(reponse(repbase)+"/"),16)
    wend
  wend
End Sub
'------------------------------------------------------------------------
Function RemplaceChaine(ByVal chaine As String, src As String, dest As String,_
     casse As Boolean)
  'Auteurs: Laurent Godard & Bernard Marcelly
  ' fournit une chaîne dont toutes les occurences de src ont été remplacées par dest
  'casse = true pour distinguer majuscules/minuscules, = false sinon
  Dim lsrc As Integer, i As Integer, kas As Integer
  dim limite as string

  limite=""
  kas = iif(casse, 0, 1)
  lsrc = len(src)
  i = instr(1, chaine, src, kas)
  while i<>0
    while i<0
      limite=limite+left(chaine,32000)
      chaine=mid(chaine,32001)
      i=instr(1, chaine, src, kas)
    wend
    ' ici i est toujours positif non nul
    if i>1 then
       limite = limite + Left(chaine, i-1) +dest
    else ' ici i vaut toujours 1
       limite = limite +dest
    endif
    ' raccourcir en deux temps car risque : i+src > 32767
    chaine = Mid(chaine, i)
    chaine = Mid(chaine, 1+lsrc)
    i = instr(1, chaine, src, kas)
```

```
  wend
  RemplaceChaine = limite + chaine
End Function
```

6Macros Calc

6.1S'agit-il d'un document tableur ?

Un document tableur est composé d'un ensemble de feuilles (sheets en anglais). Avant de pouvoir utiliser les méthodes spécifiques du tableur, vous devez disposer d'un document tableur. Vous pouvez vérifier cela comme suit :

```
Function IsSpreadhsheetDoc(oDoc) As Boolean
    On Local Error GoTo NODOCUMENTTYPE
     IsSpreadhsheetDoc =oDoc.SupportsService(_
        "com.sun.star.sheet.SpreadsheetDocument")
    NODOCUMENTTYPE:
    If Err <> 0 Then
         IsSpreadhseetDoc = False
        Resume GOON
        GOON:
    End If
End Function
```

6.2Afficher la Valeur, le Texte ou la Formule d'une cellule

```
'*********************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
Sub ExampleGetValue
    Dim oDocument As Object, oSheet As Object, oCell As Object
    oDocument=ThisComponent
    oSheet=oDocument.Sheets.getByName("Feuille1")
    oCell=oSheet.getCellByposition(0,0) 'A1
    print oCell.getValue
    'print oCell.getString
    'print oCell.getFormula
End sub
```

6.3Définir la Valeur, le Texte ou la Formule d'une cellule

```
'*********************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
Sub ExampleSetValue
    Dim oDocument As Object, oSheet As Object, oCell As Object
    oDocument=ThisComponent
    oSheet=oDocument.Sheets.getByName("Feuille1")
    oCell=oSheet.getCellByPosition(0,0) 'A1
    oCell.setValue(23658)
    'oCell..NumberFormat=2   '23658.00
    'oCell.SetString("Oups")
    'oCell.setFormula("=FUNCTION()")
    'oCell.IsCellBackgroundTransparent = TRUE
    'oCell.CellBackColor = RGB(255,141,56)
End Sub
```

6.3.1Pointer vers une Cellule dans un autre Document

Dans votre document tableur, vous pouvez accéder à une cellule dans un autre document, avec une expression du type :

```
file:///CHEMIN/NomFichier'#$Feuil1.P40
```

On peut aussi le faire dans une macro.

```
oCell = thiscomponent.sheets(0).getcellbyposition(0,0) ' A1
oCell.setFormula("=" &"'file:///home/USER/CalcFile2.sxc'#$Sheet2.K89")
```

6.4Effacer une cellule

Une liste des éléments pouvant être effacés peut être trouvée dans le SDK : com/sun/star/sheet/CellFlags.html

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub ClearDefinedRange
    Dim oDocument As Object, oSheet As Object, oSheets As Object
    Dim oCellRange As Object
    Dim nSheets As Long
        oDocument = ThisComponent
    oSheets = oDocument.Sheets
    nSheets = oDocument.Sheets.Count
    oSheet = oSheets.getByIndex(2)     Rem la plage va de 0 à n-1
    oCellRange = oSheet.getCellRangeByName("range_you_set")
    oCellRange.clearContents(_
        com.sun.star.sheet.CellFlags.VALUE | _
        com.sun.star.sheet.CellFlags.DATETIME | _
        com.sun.star.sheet.CellFlags.STRING | _
        com.sun.star.sheet.CellFlags.ANNOTATION | _
        com.sun.star.sheet.CellFlags.FORMULA | _
        com.sun.star.sheet.CellFlags.HARDATTR | _
        com.sun.star.sheet.CellFlags.STYLES | _
        com.sun.star.sheet.CellFlags.OBJECTS | _
        com.sun.star.sheet.CellFlags.EDITATTR)
End Sub
```

6.5Qu'est-ce qui est sélectionné ?

Le texte sélectionné dans un tableur peut revêtir différentes formes ; j'en comprends certaines, d'autres non.

1. Une cellule sélectionnée. Cliquez sur une cellule une fois, puis, tout en appuyant sur la touche MAJ, cliquez à nouveau sur la cellule.
2. Une partie de texte sélectionnée dans une cellule. Double cliquez dans une seule cellule, puis sélectionnez du texte.
3. Rien n'est sélectionné. Cliquez une seule fois sur une cellule ou naviguez entre plusieurs cellules.
4. Plusieurs cellules sélectionnées. Un clic unique dans une cellule, puis étirez la sélection.
5. Plusieurs cellules disjointes sélectionnées. Sélectionnez quelques cellules. Maintenez enfoncée la touche « Contrôle » et sélectionnez en quelques autres.

Jusqu'ici, je n'ai pas été en mesure de distinguer les trois premiers cas. Si j'arrive à extraire le texte sélectionné dans le cas 2, alors je peux résoudre ce problème.

```
Function CalcIsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function
    ' La sélection courante dans le contrôleur courant.
```

```
    'S'il n'y a pas de contrôleur courant, cela renvoie NULL.
    oSelections = oDoc.getCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.supportsService("com.sun.star.sheet.SheetCell") Then
        Print "Une Cellule sélectionnée = " & oSelections.getImplementationName()
        MsgBox "getString() = " & oSelections.getString()
    ElseIf oSelections.supportsService("com.sun.star.sheet.SheetCellRange") Then
        Print "Une plage de cellules sélectionnée = " & oSelections.getImplementationName()
    ElseIf oSelections.supportsService("com.sun.star.sheet.SheetCellRanges") Then
        Print "Plusieurs plages de cellules sélectionnées = " & oSelections.getImplementationName()
        Print "Count = " & oSelections.getCount()
    Else
        Print "Autre sélection = " & oSelections.getImplementationName()
    End If
End Function
```

6.6Adresse “affichable” d'une cellule

```
'Étant donnée une cellule, extraire l'adresse de la cellule sous sa forme habituelle
'D'abord, on extrait le nom de la feuille contenant la cellule.
'Ensuite, on récupère le numéro de la colonne et on le convertit en lettre.
'Enfin, on récupère le numéro de la ligne. Les lignes débutent à zéro mais sont affichées en débutant à 1.

Function PrintableAddressOfCell(the_cell As Object) As String
    PrintableAddressOfCell = "Unknown"
    If Not IsNull(the_cell) Then
        PrintableAddressOfCell = the_cell.getSpreadSheet().getName + ":" + _
            ColumnNumberToString(the_cell.CellAddress.Column) + (the_cell.CellAddress.Row+1)
    End If
End Function

' Les colonnes sont comptées en partant de 0 où 0 correspond à A
' Elles vont de A à Z, puis AA à AZ,BA à BZ,...,jusqu'à IV
' Il s'agit donc essentiellement de la façon de convertir un nombre en base 10 en un nombre
' en base 26.
' Notez que la colonne est passée en valeur (ByVal) !
Function ColumnNumberToString(ByVal the_column As Long) As String
    Dim s$
    'Enregistrez le paramètre dans une variable pour NE PAS le changer.
    'C'est un sale bug que j'ai mis du temps à trouver
    Do
        s$ = Chr(65 +  the_column MOD 26) + s$
         the_column =  the_column / 26
    Loop Until the_column = 0
    ColumnNumberToString = s$
End Function
```

6.7Insérer une date formatée dans une cellule

Cette macro insère la date dans la cellule sélectionnée. Si le document en cours n'est pas un document de tableur, un message d'erreur apparaîtra. Du code permet de formater la date selon votre choix, vous devez dé-commenter celui correspondant à votre choix :

```
'*************************************************************************
```

```
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
'utilise :  FindCreateNumberFormatStyle
Sub InsertDateIntoCell
	Dim oDesktop As Object, oController As Object, oSelection As Object
	Dim doc As Object

	oDesktop = createUnoService("com.sun.star.frame.Desktop")
	oController = oDesktop.CurrentFrame.Controller
	doc = oController.Model

	If doc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
		oSelection = oController.Selection

		' Définit la valeur de la date
		oFunction = CreateUnoService("com.sun.star.sheet.FunctionAccess")
		oFunction.NullDate = doc.NullDate
		Dim aEmpty()
		oSelection.Value = oFunction.callFunction("NOW", aEmpty())

		' Met le format de date à sa valeur par défaut
		oFormats = doc.NumberFormats
		dim aLocale as new com.sun.star.lang.Locale
		oSelection.NumberFormat = oFormats.getStandardFormat(_
			com.sun.star.util.NumberFormat.DATETIME, aLocale)

		' Met le format à une valeur complètement différente
		'oSelection.NumberFormat = FindCreateNumberFormatStyle(_
		'	"YYYYMMDD.hhmmss",  doc)
	Else
'Merci Russ Phillips pour avoir remarqué que je devais déplacer
' la commande Else vers le bas !
	MsgBox "Cette macro doit être lancée dans un document tableur"
	End If
End Sub
```

6.8Afficher la plage sélectionnée dans une boîte de dialogue

```
'*********************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
'Cette macro va prendre la plage sélectionnée et affichera une boîte de dialogue
'indiquant la plage sélectionnée et le nombre de cellules sélectionnées.
Sub SelectedCells
	oSelect=ThisComponent.CurrentSelection.getRangeAddress
	oSelectColumn=ThisComponent.CurrentSelection.Columns
	oSelectRow=ThisComponent.CurrentSelection.Rows

	CountColumn=oSelectColumn.getCount
	CountRow=oSelectRow.getCount

	oSelectSC=oSelectColumn.getByIndex(0).getName
```

```
    oSelectEC=oSelectColumn.getByIndex(CountColumn-1).getName

    oSelectSR=oSelect.StartRow+1
    oSelectER=oSelect.EndRow+1
    NoCell=(CountColumn*CountRow)

    If CountColumn=1 AND CountRow=1 Then
        MsgBox("Cellule " + oSelectSC + oSelectSR + chr(13) + "Nb Cellules = " + NoCell,, "Cellules Sélectionnées")
    Else
        MsgBox("Plage(" + oSelectSC + oSelectSR + ":" + oSelectEC + oSelectER + ")" + chr(13) + "Nb Cellules  = " +
NoCell,, "Cellules Sélectionnées")
    End If
End Sub
```

6.9Remplir la plage sélectionnée avec un texte

Cette macro simple balaye les lignes et colonnes sélectionnées en y mettant le texte “OOOPS” :

```
'*********************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
Sub FillCells
    oSelect=ThisComponent.CurrentSelection
    oColumn=oselect.Columns
    oRow=oSelect.Rows
    For nc= 0 To oColumn.getCount-1
        For nr = 0 To oRow.getCount-1
            oCell=oselect.getCellByPosition (nc,nr).setString ("OOOPS")
        Next nr
    Next nc
End Sub
```

6.10Quelques stats sur une plage sélectionnée

```
'*********************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
'Cette macro va placer des formules contenant diverses statistiques
'sur la sélection en cours, dans les cellules H7 à H15

Sub Analize
    sSum="=SUM("+GetAddress+")"
    sAverage="=AVERAGE("+GetAddress+")"
    sMin="=MIN("+GetAddress+")"
    sMax="=MAX("+GetAddress+")"
    CellPos(7,6).setString(GetAddress)
    CellPos(7,8).setFormula(sSum)
    CellPos(7,8).NumberFormat=2
    CellPos(7,10).setFormula(sAverage)
    CellPos(7,10).NumberFormat=2
    CellPos(7,12).setFormula(sMin)
    CellPos(7,12).NumberFormat=2
    CellPos(7,14).setFormula(sMax)
    CellPos(7,14).NumberFormat=2
```

```
End sub
Function GetAddress  'selected cell(s)
	oSelect=ThisComponent.CurrentSelection.getRangeAddress
	oSelectColumn=ThisComponent.CurrentSelection.Columns
	oSelectRow=ThisComponent.CurrentSelection.Rows

	CountColumn=oSelectColumn.getCount
	CountRow=oSelectRow.getCount

	oSelectSC=oSelectColumn.getByIndex(0).getName
	oSelectEC=oSelectColumn.getByIndex(CountColumn-1).getName

	oSelectSR=oSelect.StartRow+1
	oSelectER=oSelect.EndRow+1
	NoCell=(CountColumn*CountRow)

	If CountColumn=1 AND CountRow=1 then
		GetAddress=oSelectSC+oSelectSR
	Else
		GetAddress=oSelectSC+oSelectSR+":"+oSelectEC+oSelectER
	End If
End Function
Function CellPos(lColumn As Long,lRow As Long)
	CellPos= ActiveSheet.getCellByPosition (lColumn,lRow)
End Function
Function ActiveSheet
	ActiveSheet=StarDesktop.CurrentComponent.CurrentController.ActiveSheet
End Function
Sub DeleteDbRange(sRangeName As String)
	oRange=ThisComponent.DatabaseRanges
	oRange.removeByName (sRangeName)
End Sub
```

6.11Définir une plage comme plage de données

```
'*************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
Sub DefineDbRange(sRangeName As String) 'plage sélectionnée
	On Error GoTo DUPLICATENAME
	oSelect=ThisComponent.CurrentSelection.RangeAddress
	oRange=ThisComponent.DatabaseRanges.addNewByName (sRangeName,oSelect )
DUPLICATENAME:
	If Err <> 0 Then
		MsgBox("Nom dupliqué",,"INFORMATION")
	End If
End Sub
```

6.12Supprimer une plage de données

```
'*************************************************************
'Auteur : Sasa Kelecevic
'email :  scat@teol.net
```

```
Sub DeleteDbRange(sRangeName As String)
    oRange=ThisComponent.DatabaseRanges
    oRange.removeByName (sRangeName)
End Sub
```

6.13Tracer le contour d'une plage

Vous modifiez ici une structure temporaire. Utilisez quelque chose comme :

```
'*************************************************************************
'Auteur : Niklas Nebel
'email : niklas.nebel@sun.com
' Définir les bordures sous Calc
oRange = ThisComponent.Sheets(0).getCellRangeByPosition(0,1,0,63)
aBorder = oRange.TableBorder

aBorder.BottomLine = lHor
oRange.TableBorder = aBorder
```

Il dit toujours que ce qui suit ne fonctionnera pas car cela modifie une structure temporaire.

```
lHor.Color = 0: lHor.InnerLineWidth = 0: lHor.OuterLineWidth = 150:
dim lHor as New com.sun.star.table.BorderLinelHor.LineDistance = 0
ThisComponent.Sheets(0).getCellRangeByPosition(0,1,0,63).TableBorder.BottomLine = lHor
```

6.14Trier une plage

La macro suivante trie les données sur la première colonne, par ordre croissant. Par défaut, vous spécifiez les colonnes à trier, et ce sont les lignes qui sont réorganisées.

```
'*************************************************************************
'Auteur : Sasa Kelecevic
'email : scat@teol.net
Sub SortRange
    Dim oSheetDSC,oDSCRange  As Object
    Dim aSortFields(0) as new com.sun.star.util.SortField
    Dim aSortDesc(0) as new com.sun.star.beans.PropertyValue

    'définissez le nom de votre feuille
    oSheetDSC = ThisComponent.Sheets.getByName("Feuille1")

    'Définissez l'adresse de votre plage
    oDSCRange = oSheetDSC.getCellRangeByName("A1:L16")

    ThisComponent.getCurrentController.select(oDSCRange)

    aSortFields(0).Field = 0
    aSortFields(0).SortAscending = FALSE

    aSortDesc(0).Name = "SortFields"
    aSortDesc(0).Value = aSortFields()
    oDSCRange.Sort(aSortDesc())
End sub
```

Supposons que l'on veuille faire un tri sur les seconde et troisième colonnes dont la première est du texte et la seconde doit être triée numériquement. Il nous faudra deux champs de tri au lieu d'un seul.

```
Sub Main
  Dim oSheetDSC As Object, oDSCRange As Object
  Dim aSortFields(1) As New com.sun.star.util.SortField
  Dim aSortDesc(0) As New com.sun.star.beans.PropertyValue

  'Définissez le nom de votre feuille
  oSheetDSC = THISCOMPONENT.Sheets.getByName("Feuille1")

  'définissez l'adresse de votre plage
  oDSCRange = oSheetDSC.getCellRangeByName("B3:E6")
  THISCOMPONENT.getCurrentController.select(oDSCRange)

  'Un autre type de tri pourrait utiliser
  'com.sun.star.util.SortFieldType.AUTOMATIC
  'Souvenez vous que les champs partent de zéro et que donc cette procédure commence
  'à la colonne B et non à la colonne A
  aSortFields(0).Field = 1
  aSortFields(0).SortAscending = TRUE
  'On suppose qu'il n'y a pas de raison de définir le type de champ quand
  'on trie dans un tableur car il est ignoré
  'Un tableur connaît déjà le type
  'aSortFields(0).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC

  aSortFields(1).Field = 2
  aSortFields(1).SortAscending = TRUE
  aSortFields(1).FieldType = com.sun.star.util.SortFieldType.NUMERIC

  aSortDesc(0).Name = "SortFields"
  aSortDesc(0).Value = aSortFields()      ' aSortFields(0)
  oDSCRange.Sort(aSortDesc())             ' aSortDesc(0)
End Sub
```

Pour définir la première ligne comme ligne d'en tête, utilisez une autre propriété. Soyez certain d'avoir suffisamment dimensionné vos propriétés.

```
  aSortDesc(1).Name = "ContainsHeader"
  aSortDesc(1).Value = True
```

Bernard Marcelly a vérifié que la propriété « Orientation » fonctionne correctement. Pour définir l'orientation utilisez la propriété « Orientation » et définissez la valeur à l'une des suivantes :

com.sun.star.table.TableOrientation.ROWS
com.sun.star.table.TableOrientation.COLUMNS

Quand on trie en utilisant l'interface graphique, on ne peut trier plus de trois colonnes à la fois. Bernard Marcelly a noté que l'on ne peut pas contourner cette limitation en utilisant la macro. On reste limité à trois lignes ou colonnes.

6.15Trouver l'élément dupliqué

Avec un tableau trié, il est facile de trouver un élément dupliqué ! Je recherche le premier et le renvoie :

```
Function FirstDuplicate(sArray() As String) As String
    Dim i&
    FirstDuplicate = ""
    For i = LBound(sArray()) To UBound(sArray()) - 1
        If sArray(i) = sArray(i+1) Then
            FirstDuplicate = sArray(i)
            Exit For
```

```
            End If
        Next
End Function
```

6.16Afficher toutes les données d'une colonne

Tout en parcourant une cellule, et en affichant sa valeur, je souhaite afficher des informations à propos de cette cellule. Voici comment je procède :

```
Sub PrintDataInColumn (a_column As Integer)
    Dim oCells As Object, aCell As Object, oDocument As Object
    Dim oColumn As Object, oRanges As Object

    oDocument = ThisComponent
    oColumn = oDocument.Sheets(0).Columns(a_column)
    Print "Utilisation de la colonne column " + oColumn.getName
    oRanges = oDocument.createInstance("com.sun.star.sheet.SheetCellRanges")
    oRanges.insertByName("", oColumn)
    oCells = oRanges.Cells.createEnumeration
    If Not oCells.hasMoreElements Then Print "Désolé, pas de texte à afficher"
    While oCells.hasMoreElements
        aCell = oCells.nextElement
        'La fonction suivante est décrite ailleurs dans ce document !
        MsgBox PrintableAddressOfCell(aCell) + " = " + aCell.String
    Wend
End Sub
```

6.17Les Méthodes de Groupement

Ryan Nelson [ryan@aurelius-mfg.com] m'a parlé des possibilités de groupement dans Calc et m'a demandé comment les utiliser dans une macro. Il y a deux choses à garder à l'esprit. La première est que c'est la feuille qui crée et supprime les groupements, et la seconde, c'est que les paramètres doivent être corrects.

http://api.openoffice.org/docs/common/ref/com/sun/star/sheet/XSheetOutline.html

http://api.openoffice.org/docs/common/ref/com/sun/star/table/TableOrientation.html

```
Option Explicit
Sub CalcGroupingExample
    Dim oDoc As Object, oRange As Object, oSheet As Object
    oDoc = ThisComponent
    If Not oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
        MsgBox "Cette macro doit être exécutée dans un document Calc", 64, "Erreur"
    End If
    oSheet=oDoc.Sheets.getByName("Feuille1")
    ' Les paramètres sont (gauche, haut, droite, bas)
    oRange = oSheet.getCellRangeByPosition(2,1,3,2)
    'On aurait aussi pu utiliser COLUMNS ci-dessous
    oSheet.group(oRange.getRangeAddress(), com.sun.star.table.TableOrientation.ROWS)
    Print "Je viens de grouper la plage."
    oSheet.unGroup(oRange.getRangeAddress(), com.sun.star.table.TableOrientation.ROWS)
    Print "Je viens de dégrouper la plage."
End Sub
```

6.18Protéger vos données

Il est facile de protéger vos classeurs, il vous suffit de prendre vos feuilles et de les protéger. Mes essais montrent qu'aucune erreur ne se produit lorsque vous choisissez de protéger un document entier, mais cela ne protège pas le document entier.

```
Sub ProtectSpreadsheet
    dim oDoc As Object, oSheet As Object
    oDoc = ThisComponent
    oSheet=oDoc.Sheets.getByName("Feuille1")
    oSheet.protect("motdepasse")
    Print "Protect value = " & oSheet.isProtected()
    oSheet.unprotect("motdepasse")
    Print "Protect value = " & oSheet.isProtected()
End Sub
```

6.19Définir un texte d'en-tête et de pied de page

Cette macro va définir l'en-tête pour toutes les feuilles à « Feuille : <sheet_name> ». L'en-tête et le pied de page sont définis de façon identique, remplacez simplement « Header » par « Footer » dans les appels. Remerciements chaleureux à Oliver Brinzing [OliverBrinzing@t-online.de] qui a rempli les cases que j'ignorais, principalement que je devais réécrire l'en-tête dans le document.

```
Sub SetHeaderTextInSpreadSheet
    Dim oDoc as Object, oSheet as Object, oPstyle as Object, oHeader as Object
    Dim oText as Object, oCursor as Object, oField as Object
    oDoc = ThisComponent
    ' Prenez le style de la feuille sélectionnée.
    oSheet = oDoc.CurrentController.getActiveSheet
    oPstyle = oDoc.StyleFamilies.getByName("PageStyles").getByName(oSheet.PageStyle)

    ' Activez les en-têtes et partagez les !
    oPstyle.HeaderOn = True
    oPstyle.HeaderShared = True

    ' Il y a aussi un RightText et un LeftText
    oHeader = oPstyle.RightPageHeaderContent
    oText = oHeader.CenterText

    ' Vous pouvez maintenant affecter à l'objet texte la valeur que vous souhaitez
    ' Un texte simple peut être défini avec la méthode setString() de l'objet texte.
    ' Cependant, il faut utiliser un curseur pour insérer un champ comme le nom
    ' de la feuille en cours
    ' D'abord, je vais effacer le texte éventuellement existant !
    oText.setString("")
    oCursor = oText.createTextCursor()
    oText.insertString(oCursor,  "Sheet: ", False)
    ' Ceci donnera le nom de la feuille active !
    oField = oDoc.createInstance("com.sun.star.text.TextField.SheetName")
    oText.insertTextContent(oCursor, oField, False)

    ' Et maintenant, concernant la partie qui contient l'ensemble,
    ' vous devez réécrire l'objet en-tête car nous avons
    ' modifié un objet temporaire.
    oPstyle.RightPageHeaderContent = oHeader
End Sub
```

6.20Accéder à un contrôle de formulaire dans Calc via OOBasic

Laurent Godard a envoyé ce qui suit que je dois avouer ne pas avoir eu le temps de regarder et de comprendre ! Je regarderai plus tard !

Comment accéder à un contrôle de formulaire dans Calc via OOBasic :

```
Form Name= "Formul"
Control Name = "TheList"
sheet Name = "Feuille 1"
```

Je souhaite avoir la valeur de l'élément sélectionné dans "TheList". Tout est fait par une macro pour lier le changement dans une cellule. Merci à Oliver Brinzing pour m'avoir fourni un exemple :

```
oDocument = ThisComponent
oSheets = oDocument.Sheets
oSheet = oSheets.GetByName(SheetName)
oDpage = Osheet.DrawPage
oForm = oDpage.Forms(SheetName).getByName(ControlName)
Result= oForm.text
```

6.21Compter les entrées non vides dans une colonne

```
'oSheet Feuille contenant la colonne
'lCol colonne à examiner
'l_min ignorer les lignes précédentes
'l_max ignorer les lignes suivantes
'Return le nombre de cellules non vides dans la colonne
'J'ai utilisé le type "variant" pour les arguments optionnels pour contourner un bug de la version
' 1.0.2 qui ne parvient pas correctement les arguments optionnels pour les autres types.
Function NonBlankCellsInColumn (oSheet As Object, lCol&, _
        Optional l_min As Variant, Optional l_max As Variant) As Long
    Dim oCells As Object, oCell As Object, oColumn As Object
    Dim oAddr As Object, oRanges As Object
    Dim n&, lMin&, lMax&

    n = 0
    lMin = 0
    lMax = 2147483647
    If Not IsMissing(l_min) Then lMin = l_min
    If Not IsMissing(l_max) Then lMax = l_max

    oColumn = oSheet.Columns(lCol)
    oRanges = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")
    oRanges.insertByName("", oColumn)
    oCells = oRanges.Cells.createEnumeration

    Do While oCells.hasMoreElements
        oCell = oCells.nextElement
        oAddr = oCell.CellAddress
        If oAddr.Row > lMax Then Exit Do
        If oAddr.Row >= lMin And Len(oCell.String) > 0 Then n=n+1
    Loop
    NonBlankCellsInColumn = n
End Function
```

7Macro sous Writer

7.1Texte sélectionné, Qu'est-ce que c'est ?

Un texte sélectionné est essentiellement une étendue de texte, rien de plus. Après qu'une sélection ait été obtenue, il est possible d'obtenir le texte [getString()] et de définir le texte [setString()]. Bien que les chaînes de caractères (variables) soient limitées à 64K en taille, les sélections ne le sont pas. Il existe quelques occasions, parfois, où les méthodes getString() et setString() ont des résultats que je ne comprends pas. Il vaut donc, probablement, mieux utiliser un curseur pour naviguer dans le texte sélectionné et alors, utiliser les méthodes insertString() et insertControlCharacter() de l'objet Text. La documentation mentionne spécifiquement que les caractères blancs suivants sont supportés par la méthode insertString() : 'espace', 'TAB', 'CR' (qui insérera un saut de paragraphe), et 'LF' (qui insérera un retour à la ligne).

Les textes peuvent être sélectionnés manuellement de telle sorte que le curseur puisse être soit sur la gauche, soit sur la droite de la sélection. Une sélection a à la fois un début et une fin et vous ne pouvez pas savoir à l'avance laquelle est au début et laquelle est à la fin du texte sélectionné. Une méthode concernant ce problème est présentée ci-dessous :

Voir aussi :

http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextRange.html

http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRange.html

7.2Les Curseurs de Texte, Que Sont-Ils ?

Un TextCursor est un TextRange qui peut être déplacé dans un objet texte. Les déplacements standards incluent goLeft, goRight, goUp, et goDown. Le premier paramètre est un entier indiquant de combien de caractères ou de lignes bouger. Le second paramètre est un booléen indiquant si l'étendue du texte sélectionné doit être augmentée (True) ou pas. La valeur True est retournée tant que le mouvement intervient. Si un curseur a sélectionné du texte en se déplaçant à gauche et que vous voulez maintenant le déplacer à droite, vous utiliserez probablement oCursor.goRight(0, False) pour indiquer au curseur de se déplacer à droite et de ne pas sélectionner de texte. Cela ne laissera aucun texte sélectionné.

Un TextCursor a à la fois un début et une fin. Si la position de début est la même que la position de fin, alors aucun texte est sélectionné et la propriété IsCollapsed sera True.

Un TextCursor implémente des interfaces qui autorisent les déplacements et la (re)connaissance de positions spécifiques aux mots, phrases et paragraphes. Ceci peut faire gagner beaucoup de temps.

Attention	Curseur.gotoStart() et Curseur.gotoEnd() vont au début et à la fin du document même si le curseur est créé sur une sélection.

Vous devez être attentif au comportement de vos curseurs, car ils ne fonctionnent pas toujours comme dans l'interface utilisateur. Par exemple, un curseur de ligne vous permet de vous déplacer dans des lignes. Dans l'interface utilisateur, quand vous sautez à la fin de la ligne, le curseur se positionne en fin de ligne. Alors qu'avec un Curseur ***de Texte ?***, le curseur se positionne généralement au début de la ligne suivante. D'après Giuseppe Castagno [castagno@tecsa-srl.it], voici ci-dessous le comportement de la méthode gotoEndOfLine dans le type XlineCursor :

Le curseur saute de temps en temps au début de la ligne suivante, au lieu d'aller à la fin de la ligne courante. Cela semble ne se produire que quand il y a un hyperlien, ou un champ de texte, qui passent à la ligne suivante. Dans le cas d'un hyperlien, le curseur saute au début de la ligne suivante, tandis que pour un champ de texte, le curseur se positionne à la fin du champ de texte, dans la ligne suivante. Pour le champ de texte, ce comportement est conforme avec ce que l'on voit dans l'interface utilisateur, pour l'hyperlien, ce n'est pas le cas.

D'après Christoph Neumann [christoph@sun.com], c'est pour la raison suivante :

L'interface utilisateur place le curseur derrière le dernier caractère visible d'une ligne de texte. Les espaces d'un saut de ligne automatique ne sont pas pris en compte (ni affichés !), et donc ce ne sera pas la vraie fin de la ligne. Dans l'API, le curseur se place derrière le dernier vrai caractère de la ligne – ce qui correspond au même endroit que le premier caractère de la ligne suivante, dans le cas d'un saut de ligne automatique.

Voir aussi :

http://api.openoffice.org/docs/common/ref/com/sun/star/view/XViewCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XWordCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XSentenceCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XParagraphCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextRange.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRange.html

7.3Cadre de travail pour les textes sélectionnés

La plupart des problèmes utilisant une sélection de texte se ressemblent d'un point de vue abstrait (conceptuel).

```
Si rien n'est sélectionné alors
    travailler sur le document entier
sinon
    pour chaque zone de sélection
        travailler sur chaque zone
```

La partie difficile qui changera tout le temps est d'écrire une macro qui incrémentera dans une sélection ou entre deux curseurs.

7.3.1Est-ce que le texte est sélectionné ?

Les documentations stipulent que s'il n'y a pas de contrôleur courant, getCurrentSelection() retournera un null plutôt que les sélections. Je n'ai qu'une compréhension limitée de ceci mais je les prendrai au mot et vérifierai.

Si la longueur de sélection est zéro, alors rien n'est sélectionné. Je n'ai jamais vu de longueur de sélection à zéro, mais je le teste systématiquement. Si aucun texte n'est sélectionné, j'ai une sélection de longueur nulle. J'ai vu des exemples où une sélection de longueur nulle est déterminée comme suit :

```
                If Len(oSel.getString()) = 0 Then rien n'est sélectionné
```

Le problème avec ceci est qu'il est possible que le texte sélectionné contienne plus de 64K caractères. Je considère que ce n'est pas sûr. La meilleure solution est de créer un curseur texte à partir de la sélection et de tester que le début et la fin sont les mêmes.

```
                oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
                If oCursor.IsCollapsed() Then rien n'est sélectionné
```

Voici la fonction qui fera le test complet.

```
Function IsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function
    ' La sélection courante dans le contrôleur courant
    ' S'il n'y a pas de contrôleur courant, retourne null
    oSelections = oDoc.getCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.getCount() = 0 Then Exit Function
```

```
    If oSelections.getCount() > 1 Then
        IsAnythingSelected = True
    Else
        oSel = oSelections.getByIndex(0)
        oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
        If Not oCursor.IsCollapsed() Then IsAnythingSelected = True
    End If
End Function
```

7.3.2Comment obtenir une sélection ?

Obtenir une sélection est compliqué car il est possible d'avoir de multiples sélections non-contiguës. Certaines sélections peuvent être vides, et d'autres non. Le code écrit pour gérer la sélection de texte devrait gérer tous ces cas. L'exemple suivant navigue à travers toutes les sections sélectionnées et les imprime.

```
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub MultipleTextSelectionExample
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    '  La sélection courante dans le contrôleur courant.
    'S'il n'y a pas de contrôleur courant, retourne null.
    oSelections = ThisComponent.getCurrentSelection()
    If Not IsNull(oSelections) Then
        oText = ThisComponent.Text
        lSelCount = oSelections.getCount()
        For lWhichSelection = 0 To lSelCount - 1
            oSel = oSelections.getByIndex(lWhichSelection)
            MsgBox oSel.getString()
        Next
    End If
End Sub
```

Voir aussi :

http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRange.html

7.3.3Texte sélectionné, quelle fin est la bonne ?

Les sélections sont essentiellement des intervalles de texte avec un début et une fin. Bien que les sélections aient à la fois un début et une fin, quel côté du texte est celui qui est déterminé par la méthode de sélection. L'objet Text fournit des méthodes pour comparer les positions de début et de fin d'un intervalle de texte. La méthode“short compareRegionStarts (XTextRange R1, XTextRange R2)” retourne 1 si R1 débute avant R2, 0 si R1 a la même position que R2 et -1 si R1 débute après R2. La méthode “short compareRegionEnds (XTextRange R1, XTextRange R2)” retourne 1, si R1 termine avant R2, 0 si R1 termine à la même position que R2 et -1, si R1 termine derrière R2. J'utilise les deux méthodes suivantes pour trouver les positions du curseur la plus à gauche et la plus à droite d'une sélection de texte :

```
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
```

```
Function GetLeftMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getEnd()
    Else
        oRange = oSel.getStart()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goRight(0, False)
    GetLeftMostCursor = oCursor
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetRightMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getStart()
    Else
        oRange = oSel.getEnd()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goLeft(0, False)
    GetRightMostCursor = oCursor
End Function
```

Voir aussi :

http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextCursor.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XSimpleText.html
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRangeCompare.html

7.3.4Le modèle de macro pour le texte sélectionné

Cela m'a pris un long moment pour comprendre comment balayer les textes sélectionnés en utilisant des curseurs, donc, j'ai écrit plusieurs macros qui font les choses, dans ce que je considère le mauvais sens. Maintenant, j'utilise une structure de haut niveau pour faire ceci. L'idée est que si aucun texte n'est sélectionné, alors, il demande si la macro doit être lancée sur le document entier. Si la réponse est oui, alors un curseur est créé au début et à la fin du document et la macro de travail est appelée. Si du texte est sélectionné, alors chaque sélection est récupérée et un curseur est obtenu au début et à la fin de la sélection ; la macro de travail est alors appelée pour chacune des sélections.

La structure rejetée

J'ai définitivement rejeté la structure qui suit parce qu'elle est trop longue et pénible à répéter chaque fois que je voulais balayer à travers le texte. Elle est pourtant défendable. Vous pouvez préférer cette structure et choisir de l'utiliser :

```
Sub IterateOverSelectedTextFramework
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object
```

```
    oText = ThisComponent.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%
        i% = MsgBox("Aucun texte sélectionné !" + Chr(13) + _
            "Appeler la routine sur le document ENTIER ?", _
            1 OR 32 OR 256, "Attention")
        If i% <> 1 Then Exit Sub
        oLCursor = oText.createTextCursor()
        oLCursor.gotoStart(False)
        oRCursor = oText.createTextCursor()
        oRCursor.gotoEnd(False)
        CallYourWorkerMacroHere(oLCursor, oRCursor, oText)
    Else
        oSelections = ThisComponent.getCurrentSelection()
        lSelCount = oSelections.getCount()
        For lWhichSelection = 0 To lSelCount - 1
            oSel = oSelections.getByIndex(lWhichSelection)
            'Si je veux savoir si aucun texte n'est sélectionné, je peux
            ' faire la chose suivante :
            'oLCursor = oText.CreateTextCursorByRange(oSel)
            'If oLCursor.isCollapsed() Then ...
            oLCursor = GetLeftMostCursor(oSel, oText)
            oRCursor = GetRightMostCursor(oSel, oText)
            CallYourWorkerMacroHere(oLCursor, oRCursor, oText)
        Next
    End If
End Sub
```

Le modèle retenu

J'ai opté pour la création du modèle qui suit. Il retourne un tableau bi-dimensionnel des curseurs de début et de fin à partir desquels balayer le document. Ceci autorise l'utilisation d'un code minimal pour balayer les textes sélectionnés dans le document entier.

```
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
'sPrompt : Comment demander si balayage sur tout le document
'oCursors() : Contient les curseurs retournés
'Retourne true si balayage ou faux sinon
Function CreateSelectedTextIterator(oDoc As Object, sPrompt As String, oCursors()) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    CreateSelectedTextIterator = True
    oText = oDoc.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%
        i% = MsgBox("Aucun texte sélectionné!" + Chr(13) + sPrompt, _
            1 OR 32 OR 256, "Attention")
        If i% = 1 Then
            oLCursor = oText.createTextCursor()
            oLCursor.gotoStart(False)
```

```
            oRCursor = oText.createTextCursor()
            oRCursor.gotoEnd(False)
            oCursors = DimArray(0, 1)
            oCursors(0, 0) = oLCursor
            oCursors(0, 1) = oRCursor
        Else
            oCursors = DimArray()
            CreateSelectedTextIterator = False
        End If
    Else
        oSelections = ThisComponent.getCurrentSelection()
        lSelCount = oSelections.getCount()
        oCursors = DimArray(lSelCount - 1, 1)
        For lWhichSelection = 0 To lSelCount - 1
            oSel = oSelections.getByIndex(lWhichSelection)
            'Si je veux savoir si aucun texte est sélectionné, je peux
            ' faire la chose suivante :
            'oLCursor = oText.CreateTextCursorByRange(oSel)
            'If oLCursor.isCollapsed() Then ...
            oLCursor = GetLeftMostCursor(oSel, oText)
            oRCursor = GetRightMostCursor(oSel, oText)
            oCursors(lWhichSelection, 0) = oLCursor
            oCursors(lWhichSelection, 1) = oRCursor
        Next
    End If
End Function
```

La routine principale

Voici un exemple qui appelle la routine principale :

```
Sub PrintExample
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent, _
        "Imprimer les caractères pour le document entier ?", oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        PrintEachCharacterWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
```

7.3.5Comptage des Phrases

J'ai pondu tout ceci rapidement, et sans trop réfléchir. À utiliser à vos risque et périls !

```
REM Cela ne marchera probablement pas s'il y a des tableaux et équivalents, car le
REM Curseur de Phrase ne pourra pas entrer dans le(s) tableau(x), mais je ne l'ai pas
REM effectivement testé.
Sub CountSentences
 Dim vCursor as Variant 'Variant est plus sûr que Object
 Dim vSentanceCursor as Variant 'Variant est plus sûr que Object
 Dim vText As Variant
 Dim i
 vText = ThisComponent.Text
 vCursor = vText.CreateTextCursor()
 vSentanceCursor = vText.CreateTextCursor()
```

```
 'Place le curseur au début du document
 vCursor.GoToStart(False)
 Do While vCursor.gotoNextParagraph(True)
   ' A ce point, le paragraphe entier est sélectionné
   vSentanceCursor.gotoRange(vCursor.getStart(), False)
   Do While vSentanceCursor.gotoNextSentence(True) AND_
         vText.compareRegionEnds(vSentanceCursor, vCursor) >= 0
     vSentanceCursor.goRight(0, False)
     i = i + 1
   Loop
   vCursor.goRight(0, False)
 Loop
 MsgBox i, 0, "Nombre de phrases"
End Sub
```

7.3.6Afficher des caractères, un exemple simple

Cet exemple simple peut être utilisé avec le modèle ci-dessus :

```
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub PrintEachCharacterWorker(oLCursor As Object, oRCursor As Object, oText As Object)
      If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
      If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
      oLCursor.goRight(0, False)
      Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
            Print "Caractère = '" & oLCursor.getString() & "'"
            Rem Ceci fera en sorte que le texte sélectionné
            Rem ne le soit plus
            oLCursor.goRight(0, False)
      Loop
End Sub
```

7.3.7Enlever les espaces vides et les lignes, un exemple plus important

Cet ensemble de macros enlève (remplace) toutes les répétitions de caractères blancs par un seul caractère blanc. Il est facilement modifiable pour supprimer différents types de caractères blancs. Les différents types d'espaces sont classés par ordre d'importance et donc, si vous avez un espace normal suivi par un nouveau paragraphe, ce nouveau paragraphe restera et l'unique espace sera enlevé. Cela causera la suppression des espaces blancs avant et après une ligne.

Qu'est-ce qu'un espace blanc ?

En résolvant ce problème, ma première tâche a été de déterminer quels caractères sont des espaces blancs. Vous pouvez de façon évidente changer la définition du caractère blanc pour ignorer certains caractères.

```
'Normalement, ceci est fait avec une recherche dans un tableau, ce qui serait probablement
'plus rapide, mais je ne sais pas comment utiliser les déclarations (?) statiques en basic
Function IsWhiteSpace(iChar As Integer) As Boolean
      Select Case iChar
      Case 9, 10, 13, 32, 160
```

```
            IsWhiteSpace = True
        Case Else
            IsWhiteSpace = False
        End Select
End Function
```

Priorités des caractères pour la suppression

Ensuite, j'ai eu besoin de définir ce qu'il fallait enlever et ce qu'il fallait garder. J'ai choisi de faire ça avec la routine suivante :

```
'-1  signifie supprimer le caractère précédent
' 0 signifie ignorer ce caractère
' 1 signifie supprimer ce caractère
' La priorité, de la plus haute à la plus faible, est : 0, 13, 10, 9, 160, 32
Function RankChar(iPrevChar, iCurChar) As Integer
    If Not IsWhiteSpace(iCurChar) Then              'le caractère courant n'est pas un espace blanc, l'ignorer
        RankChar = 0
    ElseIf iPrevChar = 0 Then               'Début d'une ligne et le caractère courant est un espace blanc
        RankChar = 1                            '  donc supprimer l'espace blanc.
    ElseIf Not IsWhiteSpace(iPrevChar) Then    'Le caractère courant est un espace blanc mais le précédent ne l'est
pas
        RankChar = 0                            '  donc, l'ignorer.
    ElseIf iPrevChar = 13 Then                  'Le caractère précédent est un espace blanc avec la plus haute
priorité
        RankChar = 1                            '  donc supprimer le caractère courant.
    ElseIf iCurChar = 13 Then               'Le caractère courant est un espace blanc avec la plus haute priorité
        RankChar = -1                           '  donc supprimer le caractère précédent.
    ElseIf iPrevChar = 10 Then                  'Pas de nouveau paragraphe pour voir si le caractère précédent est LF
        RankChar = 1                            '  donc supprimer le caractère courant.
    ElseIf iCurChar = 10 Then               'Pas de nouveau paragraphe pour voir si le caractère courant est LF
        RankChar = -1                           '  donc supprimer le caractère précédent.
    ElseIf iPrevChar = 9 Then               'Pas de nouvelle ligne pour voir si le caractère précédent est TAB

        RankChar = 1            '  donc supprimer le caractère courant.
    ElseIf iCurChar = 9 Then                    'Pas de nouvelle ligne pour voir si le caractère courant est TAB
        RankChar = -1                           '  donc supprimer le caractère précédent.
    ElseIf iPrevChar = 160 Then             'Pas de TAB pour voir si le caractère précédent est un espace insécable
        RankChar = 1                            '  donc supprimer le caractère courant.
    ElseIf iCurChar = 160 Then                  'Pas de TAB pour voir si le caractère courant est un espace insécable
        RankChar = -1                           '  donc supprimer le caractère précédent.
    ElseIf iPrevChar = 32 Then                  'Pas d'espace insécable pour voir si le caractère précédent est un
espace
        RankChar = 1                            '  donc supprimer le caractère courant.
    ElseIf iCurChar = 32 Then               'Pas d'espace insécable pour voir si le caractère courant est un espace
        RankChar = -1                           '  donc supprimer le caractère précédent.
    Else                                        'Normalement, on ne devrait pas venir ici
        RankChar = 0                            '  donc on l'ignore tout simplement !
    End If
End Function
```

L'itérateur standard de texte sélectionné

C'est la manière standard pour décider si le travail doit être fait sur le document entier ou juste sur une portion.

```
'Enlevez toutes les occurences d'espace vide !
'Si le texte est sélectionné, alors il sera enlevé seulement de la région sélectionnée.
Sub RemoveEmptySpace
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent, _
        "TOUS les espaces vides seront enlevés du document ENTIER ?", oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        RemoveEmptySpaceWorker (oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
```

La routine de travail

C'est là où le vrai travail se passe :

```
Sub RemoveEmptySpaceWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim sParText As String, i As Integer
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub

    Dim iLastChar As Integer, iThisChar As Integer, iRank As Integer
    iLastChar = 0
    iThisChar = 0
    oLCursor.goRight(0, False)
    Do While oLCursor.goRight(1, True)
        iThisChar = Asc(oLCursor.getString())
        i = oText.compareRegionEnds(oLCursor, oRCursor)

        'Si au dernier caractère !
        'Alors toujours enlever l'espace blanc
        If i = 0 Then
            If IsWhiteSpace(iThisChar) Then oLCursor.setString("")
            Exit Do
        End If

        'Si on dépasse la fin, alors on sort de la routine
        If i < 0 Then Exit Do
        iRank = RankChar(iLastChar, iThisChar)
        If iRank = 1 Then

            'Je vais effacer ce caractère.
            'Je ne change pas iLastChar parce qu'il n'a pas changé !
            'Print "Effacer le Courant avec " + iLastChar + " et " + iThisChar
            oLCursor.setString("")
        ElseIf iRank = -1 Then

            'Cela désélectionnera le caractère sélectionné et alors en sélectionne un
            'plus à gauche.
            oLCursor.goLeft(2, True)

            'Print "Effacer avec à gauche " + iLastChar + " et " + iThisChar
            oLCursor.setString("")
            oLCursor.goRight(1, False)
            iLastChar = iThisChar
        Else
```

```
            oLCursor.goRight(0, False)
            iLastChar = iThisChar
        End If
    Loop
End Sub
```

7.3.8Supprimer les paragraphes vides, encore un autre exemple

Il est préférable de paramétrer « AutoFormat » pour supprimer les paragraphes vides, et de l'appliquer au document en question. Cliquez sur « Outils=>AutoCorrection/AutoFormat... » et choisissez l'onglet « Options ». Une des options est « Supprimer les paragraphes vides ». Vérifiez que cette entrée est cochée. Maintenant, vous pouvez appliquer l'AutoFormat (*NdT : Format-AutoFormat-Appliquer*) et tous les paragraphes vides sont supprimés.

Si vous ne voulez supprimer que les paragraphes vides sélectionnés, alors vous aurez besoin d'une macro. Si du texte est sélectionné, alors les paragraphes vides sont supprimés à l'intérieur de celui-ci. Si aucun texte n'est sélectionné, alors les paragraphes vides sont supprimés du document entier. Cette première macro se répète à travers tous les textes sélectionnés. Si aucun texte n'est sélectionné, elle crée un curseur au début et à la fin du document et travaille sur le document entier. La chose fondamentale à voir dans cette macro est comment traverser le texte basé sur les paragraphes. La macro enlevant les espaces vides est la macro la plus sûre parce qu'elle n'extrait pas de chaîne pour travailler.

```
Sub RemoveEmptyParsWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim sParText As String, i As Integer
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.gotoNextParagraph(TRUE) AND oText.compareRegionEnds(oLCursor, oRCursor) > 0

        'Oui, je sais, limité à 64K ici
        'Si nous avons un paragraphe qui fait plus de 64K
        'Alors j'ai un problème !
        sParText = oLCursor.getString()
        i = Len(sParText)
        'On ne dispose pas d'évaluation logique optimisée. Zut !
        Do While i > 0
            If (Mid(sParText,i,1) = Chr(10)) OR (Mid(sParText,i,1) = Chr(13)) Then
                i = i - 1
            Else
                i = -1
            End If
        Loop
        If i = 0 Then
            oLCursor.setString("")
        Else
            oLCursor.goLeft(0,FALSE)
        End If
    loop
End Sub
```

7.3.9Texte sélectionné, temps d'exécution et comptage de mots

Toute personne ayant étudié l'algorithmie vous dira qu'il vaut mieux un algorithme meilleur plutôt qu'une machine plus puissante. Quand j'ai commencé à écrire une macro manipulant les lignes blanches et les

espaces, je l'ai écrite en utilisant des variables de chaîne de caractères. Cela a introduit la possibilité de perdre les informations de formatage et des erreurs quand cette variable excédait 64 Ko. J'ai donc réécrit cette macro en utilisant les curseurs mais j'ai alors reçu des plaintes comme quoi elle était trop lente. Une question surgit alors : Existe-t-il un meilleur moyen ?

Recherche dans le texte sélectionné pour compter les mots

Andrew Brown, qui maintient le site http://www.darwinwars.com (contenant des informations utiles sur les macros) a posé la question pour faire une recherche dans une région sélectionnée. J'ai découvert que c'était pour compter les mots d'un document et que c'était très lent, trop lent.

Utilisation de String pour compter les mots

Le code existant comptait le nombre d'espaces dans la portion sélectionnée et se servait de ce comptage pour déterminer le nombre de mots. J'ai écrit ma propre version un peu plus généraliste, légèrement plus rapide et donnant la bonne réponse.

```
Function ADPWordCountStrings(vDoc) As String
    Rem mettre ici le caractère que l'on veut comme séparateur de mot
    Dim sSeps$
    sSeps = Chr$(9) & Chr$(13) & Chr$(10) & " ,;."
    Dim bSeps(256) As Boolean, i As Long
    For i = LBound(bSeps()) To UBound(bSeps())
        bSeps(i) = False
    Next
    For i = 1 To Len(sSeps)
        bSeps(Asc(Mid(sSeps, i, 1))) = True
    Next

    Dim nSelChars As Long, nSelwords As Long, nSel%, nNonEmptySel%, j As Long, s$
    Dim vSelections, vSel, vText, vCursor
    ' La sélection en cours dans le contrôleur courant.
    'Si il n'y a pas de contrôleur en cours, retourne Null.
    vSelections = vDoc.getCurrentSelection()
    If IsNull(vSelections) Then
        nSel = 0
    Else
        nSel = vSelections.getCount()
    End If
    nNonEmptySel = 0
    Dim lTemp As Long, bBetweenWords As Boolean, bIsSep As Boolean
    On Local Error Goto BadOutOfRange
    Do While nSel > 0
        nSel = nSel - 1
        s = vSelections.GetByIndex(nSel).getString()
        Rem Regarde si c'est une sélection vide
        lTemp = Len(s)
        If lTemp > 0 Then
            nSelChars = nSelChars + lTemp
            nNonEmptySel = nNonEmptySel + 1
            Rem Est ce que ça commence sur un mot ?
            If bSeps(Asc(Mid(s, 1, 1))) Then
                bBetweenWords = True
            Else
                bBetweenWords = False
```

```
                nSelWords = nSelWords + 1
            End If
            For j = 2 To lTemp
                bIsSep = bSeps(Asc(Mid(s, j, 1)))
                If bBetweenWords <> bIsSep Then
                    If bBetweenWords Then
                        Rem Compte un mot nouveau seulement si j'étais entre deux 2 mots
                        Rem et que je ne le suis plus
                        bBetweenWords = False
                        nSelWords = nSelWords + 1
                    Else
                        bBetweenWords = True
                    End If
                End If
            Next
        End If
    Loop
    On Local Error Goto 0

    Dim nAllChars As Long, nAllWords As Long, nAllPars As Long
    ' Accède aux statistiques du document
    nAllChars = vDoc.CharacterCount
    nAllWords = vDoc.WordCount
    nAllPars = vDoc.ParagraphCount

    Dim sRes$
    sRes = "Compteurs du document:" & chr(13) & nAllWords & "  mots. "  &_
        chr(13) & "(" & nAllChars & " caractères." & chr(13) & nAllPars &_
        " Paragraphes.)" & chr(13) & chr(13)
    If nNonEmptySel > 0 Then
        sRes = sRes  & "Compteurs du texte sélectionné:" & chr(13) & nSelWords  &_
            "  mots" & chr(13) & "(" & nSelChars  & " caractères)" &_
            chr(13) & "dans " & str(nNonEmptySel) & " sélection"
        If nNonEmptySel > 1 Then sRes = sRes & "s"
        sRes=sRes & "." & chr(13) & chr(13) & "Document minus selected:" &_
            chr(13)& str(nAllWords-nSelWords) & " mots."
    End If
    'MsgBox(sRes,64,"ADP Word Count")
    ADPWordCountStrings = sRes
    Exit Function

BadOutOfRange:
    bIsSep = False
    Resume Next
End Function
```

Chaque plage sélectionnée est extraite dans une variable string. Cette méthode échoue si la plage est supérieure à 64 K. La valeur ASCII de chaque caractère est contrôlée pour vérifier s'il doit être considéré comme un caractère de séparation de mot. C'est fait par une recherche dans un tableau. C'était efficace mais échouait s'il y avait un caractère spécial avec une valeur ASCII supérieure au tableau. Une gestion d'erreur a donc été utilisée. Un traitement spécial est effectué pour que des valeurs correctes avec différentes sélections. Cela a pris 2.7 secondes pour contrôler 8000 mots.

Utilisation d'un curseur de caractère pour compter les mots

Comme tentative de contourner la limite de 64K, j'ai écrit une version utilisant des curseurs pour traverser le texte caractère par caractère. Cette version a pris 47 secondes pour compter les 8000 mots. On utilise la même méthode que précédemment mais la surcharge pour utiliser un curseur sur chaque caractère est prohibitive.

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function ADPWordCountCharCursor(vDoc) As String
    Dim oCursors(), i%, lNumWords As Long
    lNumWords = 0
    If Not CreateSelectedTextIterator(vDoc, _
        "Compter les mots de tout le document ?", oCursors()) Then Exit Function
    For i% = LBound(oCursors()) To UBound(oCursors())
        lNumWords = lNumWords + WordCountCharCursor(oCursors(i%, 0), oCursors(i%, 1), vDoc.Text)
    Next
    ADPWordCountCharCursor = "Total des mots = " & lNumWords
End Function
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function WordCountCharCursor(oLCursor, oRCursor, vText)
    Dim lNumWords As Long
    lNumWords = 0
    WordCountCharCursor = lNumWords
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(vText) Then Exit Function
    If vText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Function

    Dim sSeps$
    sSeps = Chr$(9) & Chr$(13) & Chr$(10) & " ,;."
    Dim bSeps(256) As Boolean, i As Long
    For i = LBound(bSeps()) To UBound(bSeps())
        bSeps(i) = False
    Next
    For i = 1 To Len(sSeps)
        bSeps(Asc(Mid(sSeps, i, 1))) = True
    Next

    On Local Error Goto BadOutOfRange
    Dim bBetweenWords As Boolean, bIsSep As Boolean
    oLCursor.goRight(0, False)
    oLCursor.goRight(1, True)
    Rem Est ce que ça commence sur un mot ?
    If bSeps(Asc(oLCursor.getString())) Then
        bBetweenWords = True
    Else
        bBetweenWords = False
        lNumWords = lNumWords + 1
    End If
    oLCursor.goRight(0, False)

    Do While oLCursor.goRight(1, True) AND vText.compareRegionEnds(oLCursor, oRCursor) >= 0
```

```
            bIsSep = bSeps(Asc(oLCursor.getString()))
            If bBetweenWords <> bIsSep Then
                If bBetweenWords Then
                    Rem Compte un mot nouveau seulement si j'étais entre deux 2 mots
                    Rem et que je ne le suis plus
                    bBetweenWords = False
                    lNumWords = lNumWords + 1
                Else
                    bBetweenWords = True
                End If
            End If
            oLCursor.goRight(0, False)
        Loop
        WordCountCharCursor = lNumWords
        Exit Function

BadOutOfRange:
        bIsSep = False
        Resume Next
End Function
```

Utilisation d'un curseur de mot pour le comptage

C'est actuellement la méthode la plus rapide. Cette macro utilise un curseur de mots et laisse OOo trouver où les mots commencent et se terminent. Elle va parcourir les 8000 mots en 1.7 secondes. Cette macro avance de mot en mot, comptant combien de ruptures de mots elle trouve. Le résultat doit donc être incrémenté de 1.

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function ADPWordCountWordCursor(vDoc) As String
    Dim oCursors(), i%, lNumWords As Long
    lNumWords = 0
    If Not CreateSelectedTextIterator(vDoc, _
        "Compter les mots de tout le document ?", oCursors()) Then Exit Function
    For i% = LBound(oCursors()) To UBound(oCursors())
        lNumWords = lNumWords + WordCountWordCursor(oCursors(i%, 0), oCursors(i%, 1), vDoc.Text)
    Next
    ADPWordCountWordCursor = "Total des mots = " & lNumWords
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function WordCountWordCursor(oLCursor, oRCursor, vText)
    Dim lNumWords As Long
    lNumWords = 0
    WordCountWordCursor = lNumWords
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(vText) Then Exit Function
    If vText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Function
    oLCursor.goRight(0, False)
    Do While oLCursor.gotoNextWord(False) AND vText.compareRegionEnds(oLCursor, oRCursor) >= 0
        lNumWords = lNumWords + 1
    Loop
    WordCountWordCursor = lNumWords
```

```
End Function
```

Réflexions finales sur le comptage et le temps d'exécution

Si votre solution à un problème est trop lente, il existe peut être un autre moyen. Dans OOo, les curseurs peuvent se déplacer suivant les caractères, les mots et les paragraphes. Le curseur utilisé peut amener de grandes différences lors de l'exécution.

Si vous désirez compter le nombre de mots d'une sélection, je vous recommande de regarder le site de Andrew Brown, http://www.darwinwars.com car il y travaille activement. Aux dernières nouvelles, il abordait les choses en mettant les mots dans des tables.

7.3.10Comptage des Mots, La macro à utiliser !

La macro suivante m'a été envoyée par Andrew Brown, comme mentionné précédemment. N'hésitez pas à aller sur son site web, il y a beaucoup de choses intéressantes. Et pour ceux qui l'ignorent, il a écrit un livre. Cela ne parle pas de programmation des macros, mais j'ai beaucoup aimé certaines parties. À tester !

```
Sub acbwc
' v2.0.1
' 5 sept 2003
'  inclut les notes de bas de page et les sélections de toutes tailles
'  encore lent avec les sélections de grande taille, mais c'est de la faute d'Hamburg :-)
' v 2.0.1  un peu plus rapide, avec une meilleure routine cursorcount
'  n'est plus infiniment longue quand il y a beaucoup de notes de bas de pages.

' acb, juin 2003
'  version réécrite
' de la macro dvwc, par moi et  Daniel Vogelheim

' septembre 2003  J'ai changé le comptable en utilisant un curseur de mot pour les sélections de grande taille
'  D'après les conseils d'Andrew Pitonyak.
'  Ce n'est pas parfait, malgré tout, essentiellement parce que les déplacements mot par mot sont erratiques.
'  Cela exagère légèrement le nombre de mots dans une sélection, en comptant
'  Les sauts de paragraphe et quelques éléments de ponctuation comme des mots.
'  Mais c'est bien plus rapide que l'ancienne méthode.

Dim xDoc, xSel, nSelcount
Dim nAllChars
Dim nAllWords
Dim nAllPars
Dim thisrange, sRes
Dim nSelChars, nSelwords, nSel
Dim atext, bigtext
Dim fnotes, thisnote, nfnotes, fnotecount
Dim oddthing,startcursor,stopcursor
  xDoc = thiscomponent
  xSel = xDoc.getCurrentSelection()
  nSelCount = xSel.getCount()
  bigText=xDoc.getText()

  '  À la demande de nos auditeurs...
  fnotes=xdoc.getFootNotes()
  If fnotes.hasElements() Then
    fnotecount=0
```

```
  For nfnotes=0 To fnotes.getCount()-1
  thisnote=fnotes.getbyIndex(nfnotes)
  startcursor=thisnote.getStart()
  stopcursor=thisnote.getEnd()
  Do While thisnote.getText().compareRegionStarts(startcursor,stopcursor) AND _
        startcursor.gotoNextWord(FALSE)
    fnotecount=fnotecount+1
  Loop
'       msgbox(startcursor.getString())
'       fnotecount=fnotecount+stringcount(thisnote.getstring())
'       fnotecount=fnotecount+CursorCount(thisnote,bigtext)
  Next nfnotes
 End If

 '  Le prochain "If" résout le problème suivant : si vous aviez sélectionné du texte,
 ' puis l'avez désélectionné, et refait le comptage, alors la sélection vide était encore décomptée
 '  ce qui était trompeur et inesthétique
 If nSelCount=1 and xSel.getByIndex(0).getString()="" Then
  nSelCount=0
 End If

 '  accès aux statistiques du document
 nAllChars = xDoc.CharacterCount
 nAllWords = xDoc.WordCount
 nAllPars = xDoc.ParagraphCount

 '  mise à zéro des compteurs
 nSelChars = 0
 nSelWords = 0
 '  la partie marrante commence ici
 '  itération sur plusieurs sélections
 For nSel = 0 To nSelCount - 1
  thisrange=xSel.GetByIndex(nSel)
  atext=thisrange.getString()
  If len(atext)< 220 Then
    nselwords=nSelWords+stringcount(atext)
  Else
    nselwords=nSelWords+Cursorcount(thisrange)
  End If
  nSelChars=nSelChars+len(atext)
 Next nSel

 '  réécriture des boîtes de dialogue, pour une meilleure lisibilité
  If fnotes.hasElements() Then
    sRes="Total du document (y compris notes de bas de page) : "  + nAllWords + "  Mots. "  + chr(13)
    sRes= sRes + "Nb mots sans les notes : " + str(nAllWords-fnotecount) +_
         " mots. "  + chr(13)+"(Total: " +nAllChars +" caractères dans "
  Else
    sRes= "Total document : "  + nAllWords +"  mots. "  + chr(13)+"(" + _
         nAllChars +" caractères dans  "
  End If
  sRes=sRes + nAllPars +" Paragraphes.)"+ chr(13)+ chr(13)
  If nselCount>0 Then
```

```
    sRes=sRes  + "Texte sélectionné : " + nSelWords  + "  mots" + chr(13) +_
              "(" + nSelChars  + " caractères"
    If nSelcount=1 Then
      sRes=sRes  + " dans " + str(nselCount) + " sélection(s).)"
    Else
      REM  Je ne sais pas pourquoi, mais il faut l'ajustement suivant
      sRes=sRes +  " dans " + str(nselCount-1) +" sélection(s).)"
    End If
    sRes=sRes+chr(13)+chr(13)+"Document sans les sélections :" + chr(13)+_
      str(nAllWords-nSelWords) + " mots." +chr(13) +chr(13)
  End If
  If fnotes.hasElements() Then
    sRes=sRes+"Il y a  "+ str(fnotecount) + " mots dans "+ fnotes.getCount() +_
           " note(s) de bas de page." +chr(13) +chr(13)
  End If
  msgbox(sRes,64,"Compteur de Mots acb")
End Sub

function Cursorcount(aRange)
' acb septembre 2003
' compteur rapide pour les grandes sélections
'  fondé sur la fonction WordCountWordCursor() développée par Andrew Pitonyak
'  Mais rendue plus grossière, selon ma tendance naturelle,
Dim lnumwords as long
Dim atext
Dim startcursor, stopcursor as object
  atext=arange.getText()
  lnumwords=0
  If not atext.compareRegionStarts(aRange.getStart(),aRange.getEnd()) Then
    startcursor=atext.createTextCursorByRange(aRange.getStart())
    stopcursor=atext.createTextCursorByRange(aRange.getEnd())
  Else
    startcursor=atext.createTextCursorByRange(aRange.getEnd())
    stopcursor=atext.createTextCursorByRange(aRange.getStart())
  End If
  Do while aText.compareRegionEnds(startCursor, stopcursor) >= 0 and _
        startCursor.gotoNextWord(False)
     lnumwords=lnumwords+1
  Loop
  CursorCount=lnumwords-1
end function


Function stringcount(astring)
' acb juin 2003
'  compteur de mots plus lent, mais plus précis
'  à utiliser avec des sélections plus petites
' affûté par David Hammerton (http://crazney.net/) en septembre 2003
'  pour sauver d'un juste courroux ceux qui mettent deux espaces après les signes de ponctuation.
Dim nspaces,i,testchar,nextchar
  nspaces=0
  For i= 1 To len(astring)-1
    testchar=mid(astring,i,1)
```

```
  select Case testchar
  Case " ",chr(9),chr(13)
    nextchar = mid(astring,i+1,1)
    select Case nextchar
      Case " ",chr(9),chr(13),chr(10)
        nspaces=nspaces
      Case Else
        nspaces=nspaces+1
    end select
  end select
 Next i
 stringcount=nspaces+1
end function
```

7.4Remplacer l'espace sélectionné en utilisant des chaînes de caractères

En général, vous ne devriez pas enlever d'espace supplémentaire en lisant le texte sélectionné et en écrivant de nouvelles valeurs en retour. Une des raisons est que les chaînes de caractères sont limitées à 64K, et l'autre, qu'il est possible de perdre de l'information de formatage. J'ai laissé ces exemples en place parce qu'ils fonctionnent pour résoudre les problèmes pour lesquels ils ont été écrits avant que j'aie appris comment je pouvais faire la même chose avec les curseurs. Ils démontrent également des techniques d'insertion de caractères spéciaux. Cette première macro remplace tous les nouveaux paragraphes et nouvelles lignes avec un caractère d'espace. Ce sont aussi des exemples qui démontrent comment insérer des caractères de contrôle (CR, LF etc.) dans le texte.

```
Sub SelectedNewLinesToSpaces
    Dim lSelCount&, oSelections As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim s$, bSomethingChanged As Boolean

    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    For iWhichSelection = 0 To lSelCount - 1
        bSomethingChanged = False

        Rem et si la chaîne de caractères est plus grande que 64K ? Oups
        s = oSelections.getByIndex(iWhichSelection).getString()
        lIndex = 1
        Do While lIndex < Len(s)
            Select Case Asc(Mid(s, lIndex, 1))
            Case 13
                'Nous avons trouvé un nouveau marqueur du paragraphe.
                'Le prochain caractère sera un 10 !
                If lIndex < Len(s) And Asc(Mid(s, lIndex+1, 1)) = 10 Then
                    Mid(s, lIndex, 2, " ")
                Else
                    Mid(s, lIndex, 1, " ")
                End If
                lIndex = lIndex + 1
                bSomethingChanged = True
            Case 10
                'Nouvelle ligne à moins que le caractère antérieur soit un 13
                'Enlever cette déclaration "Case 10" pour ignorer seulement nouvelles lignes !
                If lIndex > 1 And Asc(Mid(s, lIndex-1, 1)) <> 13 Then
                    'C'est vraiment une nouvelle ligne et PAS un nouveau paragraphe.
```

```
                    Mid(s, lIndex, 1, " ")
                    lIndex = lIndex + 1
                    bSomethingChanged = True
                Else
                    'et non ! celui-ci était vraiment un nouveau paragraphe !
                    lIndex = lIndex + 1
                End If
            Case Else
                'Ne rien faire si nous ne trouvons pas quelque chose d'autre
                lIndex = lIndex + 1
            End Select
        Loop
        If bSomethingChanged Then
            oSelections.getByIndex(iWhichSelection).setString(s)
        End If
    Next
End Sub
```

Il m'a aussi été demandé de convertir de nouveaux paragraphes en nouvelles lignes. Utiliser des curseurs est clairement une meilleure idée, mais je ne savais pas comment le faire. Je pense que cet exemple est encore instructif, donc je l'ai laissé. J'efface en premier le texte sélectionné et ensuite, commence à rajouter le texte :

```
Sub SelectedNewParagraphsToNewLines
    Dim lSelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, lLastCR As Long, lLastNL As Long

    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    oText=ThisComponent.Text

    For iWhichSelection = 0 To lSelCount - 1
        oSelection = oSelections.getByIndex(iWhichSelection)
        oCursor=oText.createTextCursorByRange(oSelection)
        s = oSelection.getString()

        'Supprimer le texte sélectionné
        oCursor.setString("")
        lIndex = 1
        Do While lIndex <= Len(s)
            Select Case Asc(Mid(s, lIndex, 1)
            Case 13
                oText.insertControlCharacter(oCursor, _
        com.sun.star.text.ControlCharacter.LINE_BREAK, False)

                'J'aurais aimé avoir un booléen court
                'Passer le prochain LF s'il y en a un. Je pense
                'qu'il y en aura toujours mais je ne peux pas le vérifier.
                If (lIndex < Len(s)) Then
                    If Asc(Mid(s, lIndex+1, 1)) = 10 Then lIndex = lIndex + 1
                End If
            Case 10
```

```
                oText.insertControlCharacter(oCursor, _
                    com.sun.star.text.ControlCharacter.LINE_BREAK, False)
            Case Else
                oCursor.setString(Mid(s, lIndex, 1))
                oCursor.GoRight(1, False)
            End Select
            lIndex = lIndex + 1
        Loop
    Next
End Sub
```

7.4.1Exemples de comparaisons entre Curseurs et Chaînes

Voici quelques macros que j'ai écrites en utilisant les méthodes du curseur et ensuite, la même façon dont je les avais faites avant d'avoir ma structure :

```
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org

'Le but de cette macro est de faciliter l'utilisation de la méthode Texte<-->Tableau
'qui nécessite la supression des espaces blancs avant et après.
'Elle nécessite aussi de nouveaux paragraphes et PAS de nouvelles lignes !
Sub CRToNLMain
    Dim oCursors(), i%, sPrompt$

    sPrompt$ = "Convertir les nouveaux paragraphes en nouvelles lignes pour le document entier ?"
    If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        CRToNLWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
Sub CRToNLWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Do While oLCursor.gotoNextParagraph(False) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
        oLCursor.goLeft(1, True)
        oLCursor.setString("")
        oLCursor.goRight(0, False)
        oText.insertControlCharacter(oLCursor,_
            com.sun.star.text.ControlCharacter.LINE_BREAK, True)
    Loop
End Sub
'*************************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org

'Le but réel de cette macro est de faciliter la méthode Texte<-->Tableau
'qui nécessite la suppression des espaces blancs avant et après.
'Elle nécessite aussi de nouveaux paragraphes et PAS de nouvelles lignes !
Sub SpaceToTabsInWordsMain
    Dim oCursors(), i%, sPrompt$
```

```
    sPrompt$ = "Convertir les espaces en TABULATIONS pour le document ENTIER ?"
    If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        SpaceToTabsInWordsWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
Sub SpaceToTabsInWordsWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    Dim iCurrentState As Integer, iChar As Integer, bChanged As Boolean
    Const StartLineState = 0
    Const InWordState = 1
    Const BetweenWordState = 2

    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    iCurrentState = StartLineState
    bChanged = False
    Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
        iChar = Asc(oLCursor.getString())
        If iCurrentState = StartLineState Then
            If IsWhiteSpace(iChar) Then
                oLCursor.setString("")
            Else
                iCurrentState = InWordState
            End If
        ElseIf iCurrentState = InWordState Then
            bChanged = True
            Select Case iChar
            Case 9
                Rem Il s'agit déjà d'une tabulation, l'ignorer
                iCurrentState = BetweenWordState
            Case 32, 160

                Rem Convertit l'espace en une tabulation
                oLCursor.setString(Chr(9))
                oLCursor.goRight(1, False)
                iCurrentState = BetweenWordState
            Case 10
                Rem
                oText.insertControlCharacter(oLCursor,_
                    com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, True)
                oLCursor.goRight(1, False)
                iCurrentState = StartLineState
            Case 13
                iCurrentState = StartLineState
            Case Else
                oLCursor.gotoEndOfWord(True)
            End Select
        ElseIf iCurrentState = BetweenWordState Then
            Select Case iChar
            Case 9, 32, 160
                Rem Nous avons déjà ajouté une tabulation, chose superflue
                oLCursor.setString("")
```

```
            Case 10

                Rem Enlève la nouvelle ligne et insère un nouveau paragraphe
                Rem être certain d'effacer la TABulation de tête que nous avions déjà
                Rem ajoutée et cela devrait aller !
                oText.insertControlCharacter(oLCursor,_
                    com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, True)
                oLCursor.goLeft(0, False)
                Rem and over the TAB for deletion
                oLCursor.goLeft(1, True)
                oLCursor.setString("")
                oLCursor.goRight(1, False)
                iCurrentState = StartLineState
            Case 13

                Rem En premier, sauvegarder le CR, ensuite, sélectionner la TAB et la supprimer
                oLCursor.goLeft(0, False)
                oLCursor.goLeft(1, False)
                oLCursor.goLeft(1, True)
                oLCursor.setString("")

                Rem Enfin, revenir sur le CR que nous ignorons
                oLCursor.goRight(1, True)
                iCurrentState = StartLineState
            Case Else
                iCurrentState = InWordState
                oLCursor.gotoEndOfWord(False)
            End Select
        End If
        oLCursor.goRight(0, False)
    Loop
    If bChanged Then
        Rem Pour arriver jusqu'ici, nous sommes allés un caractère trop loin
        oLCursor.goLeft(1, False)
        oLCursor.goLeft(1, True)
        If Asc(oLCursor.getString()) = 9 Then oLCursor.setString("")
    End If
End Sub
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub TabsToSpacesMain
    Dim oCursors(), i%, sPrompt$

    sPrompt$ = "Convertir les TABS en Espaces pour le document ENTIER ?"
    If Not CreateSelectedTextIterator(ThisComponent, sPrompt$, oCursors()) Then Exit Sub
    For i% = LBOUND(oCursors()) To UBOUND(oCursors())
        TabsToSpacesWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent.Text)
    Next i%
End Sub
Sub TabsToSpacesWorker(oLCursor As Object, oRCursor As Object, oText As Object)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oText) Then Exit Sub
    If oText.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
```

```
    oLCursor.goRight(0, False)
    Do While oLCursor.goRight(1, True) AND oText.compareRegionEnds(oLCursor, oRCursor) >= 0
        If Asc(oLCursor.getString()) = 9 Then
            oLCursor.setString("    ")    'Change une tab en 4 espaces
        End If
        oLCursor.goRight(0, False)
    Loop
End Sub
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org

'sPrompt: comment demander si doit répéter sur le texte entier
'oCursors (): A les curseurs retounés
'Retourne vrai si doit répéter et faux si ne doit pas
Function CreateSelectedTextIterator(oDoc As Object, sPrompt As String, oCursors()) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    CreateSelectedTextIterator = True
    oText = oDoc.Text
    If Not IsAnythingSelected(ThisComponent) Then
        Dim i%

        i% = MsgBox("Aucun texte sélectionné !"+ Chr(13) + sPrompt, _
            1 OR 32 OR 256, "Attention")
        If i% = 1 Then
            oLCursor = oText.createTextCursor()
            oLCursor.gotoStart(False)
            oRCursor = oText.createTextCursor()
            oRCursor.gotoEnd(False)
            oCursors = DimArray(0, 1)
            oCursors(0, 0) = oLCursor
            oCursors(0, 1) = oRCursor
        Else
            oCursors = DimArray()
            CreateSelectedTextIterator = False
        End If
    Else
        oSelections = ThisComponent.getCurrentSelection()
        lSelCount = oSelections.getCount()
        oCursors = DimArray(lSelCount - 1, 1)
        For lWhichSelection = 0 To lSelCount - 1
            oSel = oSelections.getByIndex(lWhichSelection)

            'Si je veux savoir si AUCUN texte n'est sélectionné, je pourrais
            'faire la chose suivante :
            'oLCursor = oText.CreateTextCursorByRange(oSel)
            'If oLCursor.isCollapsed() Then ...
            oLCursor = GetLeftMostCursor(oSel, oText)
            oRCursor = GetRightMostCursor(oSel, oText)
            oCursors(lWhichSelection, 0) = oLCursor
```

```
                oCursors(lWhichSelection, 1) = oRCursor
            Next
        End If
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org

'oDoc est un objet Writer
Function IsAnythingSelected(oDoc As Object) As Boolean
    Dim oSelections As Object, oSel As Object, oText As Object, oCursor As Object
    IsAnythingSelected = False
    If IsNull(oDoc) Then Exit Function

    'La sélection courante dans le contrôleur courant.
    S'il n'y a pas de contrôleur courant, retourne NULL.
    oSelections = oDoc.getCurrentSelection()
    If IsNull(oSelections) Then Exit Function
    If oSelections.getCount() = 0 Then Exit Function
    If oSelections.getCount() > 1 Then
        IsAnythingSelected = True
    Else
        oSel = oSelections.getByIndex(0)
        oCursor = oDoc.Text.CreateTextCursorByRange(oSel)
        If Not oCursor.IsCollapsed() Then IsAnythingSelected = True
    End If
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetLeftMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object

    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getEnd()
    Else
        oRange = oSel.getStart()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goRight(0, False)
    GetLeftMostCursor = oCursor
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function GetRightMostCursor(oSel As Object, oText As Object) As Object
    Dim oRange As Object, oCursor As Object
    If oText.compareRegionStarts(oSel.getEnd(), oSel) >= 0 Then
        oRange = oSel.getStart()
```

```
    Else
        oRange = oSel.getEnd()
    End If
    oCursor = oText.CreateTextCursorByRange(oRange)
    oCursor.goLeft(0, False)
    GetRightMostCursor = oCursor
End Function
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
'oSelection est une sélection de texte ou un intervalle entre curseurs
'oText est l'objet texte
Function IsWhiteSpace(iChar As Integer) As Boolean
    Select Case iChar
    Case 9, 10, 13, 32, 160
        IsWhiteSpace = True
    Case Else
        IsWhiteSpace = False
    End Select
End Function



'*************************************************************
'Ici commencent les ANCIENNES macros !
'Auteur : Andrew Pitonyak
Sub ConvertSelectedNewParagraphToNewLine
    Dim lSelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, lLastCR As Long, lLastNL As Long

    'Il peut y avoir de nombreuses sélections présentes
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    oText=ThisComponent.Text

    For iWhichSelection = 0 To lSelCount - 1
        oSelection = oSelections.getByIndex(iWhichSelection)
        oCursor=oText.createTextCursorByRange(oSelection)
        s = oSelection.getString()
        oCursor.setString("")
        lLastCR = -1
        lLastNL = -1
        lIndex = 1
        Do While lIndex <= Len(s)
            Select Case Asc(Mid(s, lIndex, 1)
            Case 13
                oText.insertControlCharacter(oCursor,_
                  com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                'j'aurais aimé avoir un booléen 'short'
                'Sauter le prochain LF s'il y en a un. Je pense qu'il
                ' y en aura toujours mais je ne peux pas le vérifier.
```

```
                    If (lIndex < Len(s)) Then
                        If Asc(Mid(s, lIndex+1, 1)) = 10 Then lIndex = lIndex + 1
                    End If
                Case 10
                    oText.insertControlCharacter(oCursor,_
                      com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                Case Else
                    oCursor.setString(Mid(s, lIndex, 1))
                    oCursor.GoRight(1, False)
                End Select
                lIndex = lIndex + 1
            Loop
        Next
End Sub

'J'ai décidé d'écrire ceci comme une machine d'états finis.
'Les machines d'états finis sont de merveilleuses choses :)
Sub ConvertSelectedSpaceToTabsBetweenWords
    Dim lSelCount&, oSelections As Object, oSelection As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim oText As Object, oCursor As Object
    Dim s$, lLastCR As Long, lLastNL As Long

    Rem Quels états sont supportés
    Dim iCurrentState As Integer
    Const StartLineState = 0
    Const InWordState = 1
    Const BetweenWordState = 2

    Rem Points de transition
    Dim iWhatFound As Integer
    Const FoundWhiteSpace = 0
    Const FoundNewLine = 1
    Const FoundOther = 2
    Const ActionIgnoreChr = 0
    Const ActionDeleteChr = 1
    Const ActionInsertTab = 2

    Rem Définir les états de transition
    Dim iNextState(0 To 2, 0 To 2, 0 To 1) As Integer
    iNextState(StartLineState, FoundWhiteSpace, 0) = StartLineState
    iNextState(StartLineState, FoundNewLine, 0)        = StartLineState
    iNextState(StartLineState, FoundOther, 0)      = InWordState

    iNextState(InWordState, FoundWhiteSpace, 0)     = BetweenWordState
    iNextState(InWordState, FoundNewLine, 0)    = StartLineState
    iNextState(InWordState, FoundOther, 0)         = InWordState

    iNextState(BetweenWordState, FoundWhiteSpace, 0)= BetweenWordState
    iNextState(BetweenWordState, FoundNewLine, 0)  = StartLineState
    iNextState(BetweenWordState, FoundOther, 0)    = InWordState

    Rem Définir les états d'action
```

```
iNextState(StartLineState, FoundWhiteSpace, 1) = ActionDeleteChr
iNextState(StartLineState, FoundNewLine, 1)        = ActionIgnoreChr
iNextState(StartLineState, FoundOther, 1)      = ActionIgnoreChr

iNextState(InWordState, FoundWhiteSpace, 1)      = ActionDeleteChr
iNextState(InWordState, FoundNewLine, 1)      = ActionIgnoreChr
iNextState(InWordState, FoundOther, 1)        = ActionIgnoreChr

iNextState(BetweenWordState, FoundWhiteSpace, 1)= ActionDeleteChr
iNextState(BetweenWordState, FoundNewLine, 1)  = ActionIgnoreChr
iNextState(BetweenWordState, FoundOther, 1)    = ActionInsertTab

'Il peut y avoir des sélections multiples présentes !
oSelections = ThisComponent.getCurrentSelection()
lSelCount = oSelections.getCount()
oText=ThisComponent.Text

For iWhichSelection = 0 To lSelCount - 1
    oSelection = oSelections.getByIndex(iWhichSelection)
    oCursor=oText.createTextCursorByRange(oSelection)
    s = oSelection.getString()
    oCursor.setString("")
    lLastCR = -1
    lLastNL = -1
    lIndex = 1
    iCurrentState = StartLineState
    Do While lIndex <= Len(s)
        Select Case Asc(Mid(s, lIndex, 1))
        Case 9, 32, 160
            iWhatFound = FoundWhiteSpace
        Case 10
            iWhatFound = FoundNewLine
            lLastNL = lIndex
        Case 13
            iWhatFound = FoundNewLine
            lLastCR = lIndex
        Case Else
            iWhatFound = FoundOther
        End Select
        Select Case iNextState(iCurrentState, iWhatFound, 1)
        Case ActionDeleteChr
            'En choisissant de ne pas insérer, il est effacé !
        Case ActionIgnoreChr
            'Cela veut dire vraiment que je dois ajouter le caractère en arrière !
            If lLastCR = lIndex Then
                'Insérer un caractère du contrôle paraît déplacer le
                'curseur autour
                oText.insertControlCharacter(oCursor,_
                  com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
                oText.insertControlCharacter(oCursor,_
                  com.sun.star.text.ControlCharacter.APPEND_PARAGRAPH, False)
                'oCursor.goRight(1, False)
                'Print "CR inséré"
```

```
                ElseIf lLastNL = lIndex Then
                    If lLastCR + 1 <> lIndex Then
                        oText.insertControlCharacter(oCursor,_
                          com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
                          'com.sun.star.text.ControlCharacter.LINE_BREAK, False)
                        'oCursor.goRight(1, False)
                        'Print "NL insérée"
                    End If
                    'Ignorer celui-ci
                Else
                    oCursor.setString(Mid(s, lIndex, 1))
                    oCursor.GoRight(1, False)
                    'Print "Quelque chose insérée"
                End If
            Case ActionInsertTab
                oCursor.setString(Chr$(9) + Mid(s, lIndex, 1))
                oCursor.GoRight(2, False)
                'Print "Tabulation insérée"
            End Select
            lIndex = lIndex + 1
            'MsgBox "index = " + lIndex + Chr(13) + s
            iCurrentState = iNextState(iCurrentState, iWhatFound, 0)
        Loop
    Next
End Sub


Sub ConvertAllTabsToSpace
    DIM oCursor As Object, oText As Object
    Dim nSpace%, nTab%, nPar%, nRet%, nTot%
    Dim justStarting As Boolean

    oText=ThisComponent.Text                'Récupèrer le composant texte
    oCursor=oText.createTextCursor()  'Créer un curseur dans le texte
    oCursor.gotoStart(FALSE)            'Aller au début mais NE PAS sélectionner le texte en allant
    Do While oCursor.GoRight(1, True) 'Bouger à droite d'un caractère et le sélectionner
        If Asc(oCursor.getString()) = 9 Then
            oCursor.setString("    ")     'Change une tabulation en 4 espaces
        End If
        oCursor.goRight(0,FALSE)            'Désélectionne le texte !
    Loop
End Sub

Sub ConvertSelectedTabsToSpaces
    Dim lSelCount&, oSelections As Object
    Dim iWhichSelection As Integer, lIndex As Long
    Dim s$, bSomethingChanged As Boolean

    'Il peut y avoir des sélections multiples présentes !
    'Il y en aura probablement une de plus qu'attendu parce que
    'il comptera l'emplacement du curseur courant comme un morceau
    'de texte sélectionné, soyez avertis !
    oSelections = ThisComponent.getCurrentSelection()
```

```
    lSelCount = oSelections.getCount()
    'Print "total selectionné = " + lSelCount
    For iWhichSelection = 0 To lSelCount - 1
        bSomethingChanged = False
        s = oSelections.getByIndex(iWhichSelection).getString()
        'Print "Le groupe de texte " + iWhichSelection + " est de taille " + Len(s)
        lIndex = 1
        Do While lIndex < Len(s)
            'Print "ascii à " + lIndex + " = " + Asc(Mid(s, lIndex, 1))
            If Asc(Mid(s, lIndex, 1)) = 9 Then
                s = ReplaceInString(s, lIndex, 1, "    ")
                bSomethingChanged = True
                lIndex = lIndex + 3
            End If
            lIndex = lIndex + 1
            'Print ":" + lIndex + "(" + s + ")"
        Loop
        If bSomethingChanged Then
            oSelections.getByIndex(iWhichSelection).setString(s)
        End If
    Next
End Sub

Function ReplaceInString(s$, index&, num&, replaces$) As String
    If index <= 1 Then
        '
        If num < 1 Then
            ReplaceInString = replaces + s
        ElseIf num > Len(s) Then
            ReplaceInString = replaces
        Else
            ReplaceInString = replaces + Right(s, Len(s) - num)
        End If
    ElseIf index + num > Len(s) Then
        ReplaceInString = Left(s,index - 1) + replaces
    Else
        ReplaceInString  = Left(s,index - 1) + replaces + Right(s, Len(s) - index - num + 1)
    End If
End Function
```

7.5Définir les attributs de texte

Quand cette macro est lancée, elle affecte le paragraphe contenant le curseur. La police et la taille sont définies. L'attribut CharPosture contrôle l'italique, CharWeight contrôle le gras, et CharUnderline contrôle le type de soulignement. Les valeurs valides se trouvent à :

http://api.openoffice.org/docs/common/ref/com/sun/star/style/CharacterProperties.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontWeight.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontSlant.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontUnderline.html

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
Sub SetTextAttributes
```

```
    Dim document As Object
    Dim Cursor
    Dim oText As Object
    Dim mySelection As Object
    Dim Font As String

    document=ThisComponent
    oText = document.Text
    Cursor = document.currentcontroller.getViewCursor()
    mySelection = oText.createTextCursorByRange(Cursor.getStart())
    mySelection.gotoStartOfParagraph(false)
    mySelection.gotoEndOfParagraph(true)
    mySelection.CharFontName="Courier New"
    mySelection.Charheight="10"
    'Il est temps de définir l'italique ou pas, selon le cas avec
    'NONE, OBLIQUE, ITALIC, DONTKNOW, REVERSE_OBLIQUE, REVERSE_ITALIC
    mySelection.CharPosture = com.sun.star.awt.FontSlant.ITALIC
    'Alors, vous voulez un texte en gras ?
    'DONTKNOW, THIN, ULTRALIGHT, LIGHT, SEMILIGHT,
    'NORMAL, SEMIBOLD, BOLD, ULTRABOLD, BLACK
    'Ces dernières sont vraiment des constantes avec THIN à 50, NORMAL à 100
    ' BOLD à 150, et BLACK à 200.
    mySelection.CharWeight = com.sun.star.awt.FontWeight.BOLD
    'Si le soulignement est votre tasse de thé
    'NONE, SINGLE, DOUBLE, DOTTED, DONTKNOW, DASH, LONGDASH,
    'DASHDOT, DASHDOTDOT, SMALLWAVE, WAVE, DOUBLEWAVE, BOLD,
    'BOLDDOTTED, BOLDDASH, BOLDLONGDASH, BOLDDASHDOT,
    'BOLDDASHDOTDOT, BOLDWAVE
    mySelection.CharUnderline = com.sun.star.awt.FontUnderline.SINGLE
    'Je n'ai pas assez expérimenté ce qui suit pour en connaître les réelles
    'implications mais je sais que cela semble définir
    'la localisation des caractères à Allemand.
    Dim aLanguage As New com.sun.star.lang.Locale
    aLanguage.Country = "de"
    aLanguage.Language = "de"
    mySelection.CharLocale = aLanguage
End Sub
```

7.6Insérer du texte

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub InsertSimpleText
    Dim oDocument As Object
    Dim oText As Object
    Dim oViewCursor As Object
    Dim oTextCursor As Object

    oDocument = ThisComponent
    oText = oDocument.Text
    oViewCursor = oDocument.CurrentController.getViewCursor()
    oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())
    ' Place le texte à insérer ici
```

```
    oText.insertString(oTextCursor, "—", FALSE)
End Sub
```

7.7 Les champs

7.7.1Insérer un champ de date formaté dans un document texte

Ceci va insérer le texte “Aujourd'hui est le <date> ” où la date est formatée selon “DD. MMM YYYY”. Cela créera le format de date s'il n'existe pas. Pour plus d'informations sur les formats valides, consultez l'aide aux rubriques « formats numériques ; formats ».

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
'utilise :  FindCreateNumberFormatStyle
Sub InsertDateField
    Dim oDocument As Object
    Dim oText As Object
    Dim oViewCursor As Object
    Dim oTextCursor As Object
    Dim oDateTime As Object

    oDocument = ThisComponent
    If oDocument.SupportsService("com.sun.star.text.TextDocument") Then
        oText = oDocument.Text
        oViewCursor = oDocument.CurrentController.getViewCursor()
        oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())
        oText.insertString(oTextCursor, "Aujourd'hui est le ", FALSE)
        ' Crée le type DateTime.
        ODateTime = oDocument.createInstance("com.sun.star.text.TextField.DateTime")
        oDateTime.IsFixed = TRUE
        oDateTime.NumberFormat = FindCreateNumberFormatStyle(_
            "DD. MMMM YYYY", oDocument)
        oText.insertTextContent(oTextCursor,oDateTime,FALSE)
        oText.insertString(oTextCursor," ",FALSE)    Else
        MsgBox "Désolé, cette macro nécessite un document texte"
    End If
End Sub
```

7.7.2 Insérer une Note (Annotation)

```
Sub AddNoteAtCursor
 Dim vDoc
 Dim vViewCursor
 Dim vCursor
 Dim vTextField

 ' Allez, on va prétendre l'avoir ajoutée il y a 10 jours !
 'http://api.openoffice.org/docs/common/ref/com/sun/star/util/Date.html
 Dim aDate As New com.sun.star.util.Date
 With aDate
   .Day   = Day(Now - 10)
   .Month = Month(Now - 10)
   .Year  = Year(Now - 10)
 End With
```

```
 vDoc = ThisComponent
 vViewCursor = vDoc.getCurrentController().getViewCursor()
 vCursor = vDoc.getText().createTextCursorByRange(vViewCursor.getStart())
 ' *** ?! Positionne le curseur invisible à la position courante.

 'http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/Annotation.html
 vTextField = vDoc.createInstance("com.sun.star.text.TextField.Annotation")
 With vTextField
   .Author  = "AP"
   .Content = "Qu'est-ce que c'est marrant d'insérer des notes dans mon document"
   ' Si vous ne la mentionnez pas, la date est celle d'aujourd'hui !
   .Date    = aDate
 End With
 vDoc.Text.insertTextContent(vCursor, vTextField, False)
End Sub
```

7.8Insérer une nouvelle page

Dans ma recherche pour insérer une nouvelle page dans un document, je suis tombé sur le lien suivant :

http://api.openoffice.org/docs/common/ref/com/sun/star/style/ParagraphProperties.html

qui discute de deux propriétés. Le *PageNumberOffset* stipule : “Si une propriété de saut de page est définie sur un paragraphe, cette propriété contient la nouvelle valeur pour le numéro de page”. La propriété *PageDescName* stipule : “Si cette propriété est définie, elle crée un saut de page avant le paragraphe auquel il appartient et assigne la valeur comme étant le nom du style de la nouvelle page à utiliser”. J'ai réfléchi que si je définissais PageDescName, alors je pourrais créer une nouvelle page et définir le numéro de page. Ce qui n'était pas dit est que PageDescName est le nom du style pour la nouvelle page à utiliser après le saut de page. Si vous n'utilisez pas un style de page existant, alors, cela ne fonctionnera pas !

```
Sub ExampleNewPage
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    oText = ThisComponent.Text
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    For lWhichSelection = 0 To lSelCount - 1
        oSel = oSelections.getByIndex(lWhichSelection)
        oLCursor = oText.CreateTextCursorByRange(oSel)
        oLCursor.gotoStartOfParagraph(false)
        oLCursor.gotoEndOfParagraph(true)

        Rem Conserve le style de page existant
        oLCursor.PageDescName = oLCursor.PageStyleName
        oLCursor.PageNumberOffset = 7
    Next
End Sub
```

7.9Gérer le style de page du document

Le style de page est paramétré en modifiant le nom de description de page. C'est très similaire au fait de démarrer une nouvelle page.

```
Sub SetDocumentPageStyle
    Dim oCursor As Object
    oCursor = ThisComponent.Text.createTextCursor()
    oCursor.gotoStart(False)
    oCursor.gotoEnd(True)
    Print "Style courant = " & oCursor.PageStyleName
    oCursor.PageDescName = "Wow"
End Sub
```

7.10Insérer un objet OLE

Le rumeur dit qu'avec OpenOffice 1.1, le code suivant insérera un objet OLE dans un document Texte. Le CLSID peut être un objet externe.

```
obj = ThisComponent.createInstance("com.sun.star.text.TextEmbeddedObject")
obj.CLSID = "47BBB4CB-CE4C-4E80-A591-42D9AE74950F"
obj.attach( ThisComponent.currentController().Selection.getByIndex(0) )
```

Si vous avez sélectionné un objet embarqué dans Writer, vous pouvez accéder à son API avec :

```
oModel = ThisComponent.currentController().Selection.Model
```

Ceci fournit la même interface à l'objet que si vous l'aviez créé en chargeant un document avec loadComponentFromURL

7.11Paramétrer le style de paragraphe

Différents styles peuvent être directement réglés en sélectionnant le texte concerné.

```
Option Explicit
Sub SetParagraphStyle
    Dim oSelections As Object, oSel As Object, oText As Object
    Dim lSelCount As Long, lWhichSelection As Long
    Dim oLCursor As Object, oRCursor As Object

    oText = ThisComponent.Text
    oSelections = ThisComponent.getCurrentSelection()
    lSelCount = oSelections.getCount()
    For lWhichSelection = 0 To lSelCount - 1
        oSel = oSelections.getByIndex(lWhichSelection)
        oSel.ParaStyleName = "Heading 2"
    Next
End Sub
```

L'exemple suivant mettra tous les paragraphes au même style :

```
'Auteur : Marc Messeant
'email : marc.liste@free.fr
Sub AppliquerStyle()
    Dim oDocument As Object, oText As Object, oViewCursor as Object, oTextCursor As Object
    oDocument   = ThisComponent
    oText = oDocument.Text

    oViewCursor = oDocument.CurrentController.getViewCursor()
    oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart())

    While oText.compareRegionStarts(oTextCursor.getStart(),oViewCursor.getEnd()) = 1
        oTextCursor.paraStyleName = "YourStyle"
        oTextCursor.gotoNextParagraph(false)
    Wend
```

```
End Sub
```

7.12Créer votre propre style

Je n'ai pas testé ce code par manque de temps, mais je pense qu'il marche :

```
vFamilies = ThisComponent.StyleFamilies
vStyle =ThisComponent.createInstance("com.sun.star.style.ParagraphStyle")
vParaStyles = vFamilies.getByName("ParagraphStyles")
vParaStyles.insertByName("MyStyle",vStyle)
```

7.13Rechercher et remplacer

Un document susceptible de recherches supporte la capacité à créer un descripteur de recherche. Il sera également possible de trouver le premier, le suivant, et toutes les occurrences du texte recherché. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/util/XSearchable.html
Chercher est assez simple et un exemple devrait fournir suffisamment d'explications :

```
Sub SimpleSearchExample
    Dim vDescriptor, vFound
    ' Création d'un descripteur depuis un document susceptible de recherches
    vDescriptor = ThisComponent.createSearchDescriptor()
    ' Indiquer le texte à chercher et autre
    ' http://api.openoffice.org/docs/common/ref/com/sun/star/util/SearchDescriptor.html
    With vDescriptor
        .SearchString = "hello"
        ' Tout ceci est "false" par défaut
        .SearchWords = true
        .SearchCaseSensitive = False
    End With
    ' Chercher le premier
    vFound = ThisComponent.findFirst(vDescriptor)
    Do While Not IsNull(vFound)
        Print vFound.getString()
        vFound.CharWeight = com.sun.star.awt.FontWeight.BOLD
        vFound = ThisComponent.findNext( vFound.End, vDescriptor)
    Loop
```

L'objet retourné par findFirst et findNext se comporte comme un curseur et la plupart des choses qu'il est possible d'effectuer avec un curseur, comme le paramétrage des attributs, sont également faisables avec cet objet.

7.13.1Remplacer du texte

Remplacer du texte est semblable à la recherche à ceci près que cela doit supporter :
http://api.openoffice.org/docs/common/ref/com/sun/star/util/XReplaceable.html
La seule méthode utile fournie par ceci : un document susceptible de recherche n'est pas capable de remplacer toutes les occurrences du texte recherché par autre chose. L'idée était que vous en cherchiez une à la fois, et vous pouviez modifier chaque occurrence manuellement. L'exemple suivant recherche le texte et remplace des choses telles que l'"a@" avec le caractère unicode 257.

```
'Auteur :  Birgit Kellner
'email : birgit.kellner@univie.ac.at
Sub AtToUnicode
    'Andy dit que dans le futur, ils devront peut-être être de type Variant pour travailler avec Array()
    Dim numbered(5) As String, accented(5) As String
    Dim n as long
```

```
    Dim oDocument as object, oReplace as object
    numbered() = Array("A@", "a@", "I@", "i@", "U@", "u@", "Z@", "z@", "O@", "o@", "H@",_
        "h@", "D@", "d@", "L@", "l@", "M@", "m@", "G@", "g@", "N@", "n@", "R@", "r@",_
        "Y@", "y@", "S@", "s@", "T@", "t@", "C@", "c@", "j@", "J@")
    accented() = Array(Chr$(256), Chr$(257), Chr(298), Chr$(299), Chr$(362), Chr$(363),_
        Chr$(377), Chr$(378), Chr$(332), Chr$(333), Chr$(7716), Chr$(7717), Chr$(7692),_
        Chr$(7693), Chr$(7734), Chr$(7735), Chr$(7746), Chr$(7747), Chr$(7748), Chr$(7749),_
        Chr$(7750), Chr$(7751), Chr$(7770), Chr$(7771), Chr$(7772), Chr$(7773), Chr$(7778),_
        Chr$(7779), Chr$(7788), Chr$(7789), Chr$(346), Chr$(347), Chr$(241), Chr$(209))
    oReplace = ThisComponent.createReplaceDescriptor()
    oReplace.SearchCaseSensitive = True
    For n = lbound(numbered()) To ubound(accented())
        oReplace.SearchString = numbered(n)
        oReplace.ReplaceString = accented(n)
        ThisComponent.ReplaceAll(oReplace)
    Next n
End Sub
```

7.13.2Chercher le texte sélectionné

Le « truc » pour chercher seulement une plage de texte sélectionnée est basée sur le fait que le curseur peut être utilisé dans la routine findNext. Vous pouvez alors chercher les points finaux de la sélection pour éviter que la recherche n'aille trop loin. Ceci vous autorise à démarrer la recherche à n'importe quelle position du curseur. La méthode findFirst n'est pas nécessaire si vous avez un objet de type curseur à qui spécifier la position du début de la recherche avec findNext. L'exemple ci dessous utilise mon système basé sur le texte sélectionné et contient quelques améliorations suggérées par Bernard Marcelly.

Voyez également :
http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextRangeCompare.html

```
'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
Sub SearchSelectedText
    Dim oCursors(), i%
    If Not CreateSelectedTextIterator(ThisComponent, _
        "Search text in the entire document?", oCursors()) Then Exit Sub
    For i% = LBound(oCursors()) To UBound(oCursors())
        SearchSelectedWorker(oCursors(i%, 0), oCursors(i%, 1), ThisComponent)
    Next i%
End Sub

'*************************************************************
'Auteur : Andrew Pitonyak
'email :   andrew@pitonyak.org
Sub SearchSelectedWorker(oLCursor, oRCursor, oDoc)
    If IsNull(oLCursor) Or IsNull(oRCursor) Or IsNull(oDoc) Then Exit Sub
    If oDoc.Text.compareRegionEnds(oLCursor, oRCursor) <= 0 Then Exit Sub
    oLCursor.goRight(0, False)
    Dim vDescriptor, vFound
    vDescriptor = oDoc.createSearchDescriptor()
    With vDescriptor
        .SearchString = "Paragraphe"
        .SearchCaseSensitive = False
```

```
    End With
    ' Il n'y a pas lieu d'éxécuter findFirst.
    vFound =  oDoc .findNext(oLCursor, vDescriptor)
    Rem Voudriez vous arrêter l'évaluation ?
    Do While Not IsNull(vFound)
        Rem Si vFound.hasElements() est vide alors on sort
        'Voyons si nous recherchons après la fin
        If -1 = oDoc.Text.compareRegionEnds(vFound, oRCursor) Then Exit Do
        Print vFound.getString()
        vFound = ThisComponent.findNext( vFound.End, vDescriptor)
    Loop
End Sub
```

7.13.3Recherches et remplacements complexes

```
'effacer du texte entre délimiteurs est actuellement aisé
Sub deleteTextBetweenDlimiters
    Dim vOpenSearch, vCloseSearch   'Ouvrir et fermer les délimiteurs
    Dim vOpenFound, vCloseFound         'Ouvrir et fermer les objets trouvés

    ' Création d'un descripteur depuis le document suceptible de recherches.
    vOpenSearch = ThisComponent.createSearchDescriptor()
    vCloseSearch = ThisComponent.createSearchDescriptor()

    ' Indiquer le texte à chercher et autre
    ' http://api.openoffice.org/docs/common/ref/com/sun/star/util/SearchDescriptor.html
    vOpenSearch.SearchString = "["
    vCloseSearch.SearchString = "]"

    ' Trouver et ouvrir le premier délimiteur
    vOpenFound = ThisComponent.findFirst(vOpenSearch)
    Do While Not IsNull(vOpenFound)

        'Rechercher le délimiteur fermant le plus proche du début
        vCloseFound = ThisComponent.findNext( vOpenFound.End, vCloseSearch)
        If IsNull(vCloseFound) Then
            Print "Trouvé une parenthèse ouvrante mais aucune parenthèse fermante !"
            Exit Do
        Else
            ' Dégager la parenthèse ouvrante sous peine de finir en haut
            ' Seulement avec le texte entre parenthèses
            vOpenFound.setString("")
            ' Selection du texte entre parenthèses
            vOpenFound.gotoRange(vCloseFound, True)
            Print "Trouve " & vOpenFound.getString()
            ' Dégager le texte concerné
            vOpenFound.setString("")
            ' Dégager la parenthèse fermante
            vCloseFound.setString("")
            ' Voulez vous vraiment tout effacer dans cet espace
            ' Alors, allons y !
            If vCloseFound.goRight(1, True) Then
                If vCloseFound.getString() = " " Then vCloseFound.setString("")
            End If
```

```
            vOpenFound = ThisComponent.findNext( vOpenFound.End, vOpenSearch)
        End If
    Loop
End Sub
```

7.13.4 Rechercher et Remplacer avec des Attributs et des Expressions régulières

Cette macro encadre tous les éléments en **GRAS** avec des accolades “{{ }}” et change l'attribut du **Gras** en Normal. Une expression régulière est utilisée pour spécifier le texte dans lequel chercher.

```
Sub ReplaceFormatting
 'code original : Alex Savitsky
 'modifié par : Laurent Godard
 ' Le but de cette macro est d'encadrer tous les éléments en GRAS par {{ }}
 ' et de changer l'attribut Gras en NORMAL
 ' Ceci se fait avec des expressions régulières
 ' Les styles doivent aussi être pris en compte

 Dim oDocument As Object
 Dim oReplace As Object
 Dim SrchAttributes(0) as new com.sun.star.beans.PropertyValue
 Dim ReplAttributes(0) as new com.sun.star.beans.PropertyValue

 oDocument = ThisComponent
 oReplace = oDocument.createReplaceDescriptor

 oReplace.SearchString = ".*"          'Expression régulière. Prendre tous les caractères
 oReplace.ReplaceString = "{{ & }}"    ' Attention au &. Place le texte trouvé en arrière plan
 oReplace.SearchRegularExpression=True  'Utiliser les expressions régulières
 oReplace.searchStyles=True            ' Nous voulons rechercher dans les styles
 oReplace.searchAll=True               ' Pour tout le document

 REM Voilà l'attribut à trouver
 SrchAttributes(0).Name = "CharWeight"
 SrchAttributes(0).Value =com.sun.star.awt.FontWeight.BOLD

 REM Voilà l'attribut par lequel remplacer le premier
 ReplAttributes(0).Name = "CharWeight"
 ReplAttributes(0).Value =com.sun.star.awt.FontWeight.NORMAL

 REM  Place les attributs dans le descripteur de remplacement
 oReplace.SetSearchAttributes(SrchAttributes())
 oReplace.SetReplaceAttributes(ReplAttributes())

 REM  Allez au boulot !
 oDocument.replaceAll(oReplace)
End Sub
```

7.14Changer la casse des mots

OOo détermine la casse d'un mot en se basant sur les propriétés de caractère. En théorie, cela signifie que l'on peut sélectionner le document entier et paramétrer la casse. Dans la pratique toutefois, les portions sélectionnées peuvent ne pas supporter la propriété casse de caractère. Comme compromis entre vitesse et problèmes possibles, j'ai choisi d'utiliser un curseur de mot pour traverser le texte

paramétrant la propriété casse de chaque mot individuellement. J'ai écrit cette macro pour travailler sur des mots entiers, c'est une décision arbitraire. Si elle ne vous convient pas, changez-la. J'ai utilisé mon système de texte sélectionné, vous aurez donc besoin de ces macros pour ces travaux.

S'il se trouve que vous avez du texte qui ne supporte pas le paramétrage de la casse de caractère, vous pouvez éviter l'apparition d'erreur en ajoutant “On Local Error Resume Next” à SetWordCase().

Attention	Paramétrer la casse ne change pas le caractère mais seulement son affichage. Si vous paramétrez un bas de casse, vous ne pourrez pas passer en haut de casse manuellement.
Attention	Dans OOo 1.0.3.1, la casse des titres est perturbée : “heLLo” devient “HeLLo”.

Voir également :
http://api.openoffice.org/docs/common/ref/com/sun/star/style/CaseMap.html

```
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Sub ADPSetWordCase()
    Dim vCursors(), i%, sMapType$, iMapType%
    iMapType = -1
    Do While iMapType < 0
        sMapType = InputBox("Quelle casse vais-je utiliser ?" & chr(10) &_
            "Rien, MAJUSCULE, minuscule, Titre ou  petites majuscules ?", "Changer le Type de Casse", "Titre")
        sMapType = UCase(Trim(sMapType))
        If sMapType = "" Then sMapType = "EXIT"
        Select Case sMapType
        Case "EXIT"
            Exit Sub
        Case "NONE"
            iMapType = com.sun.star.style.CaseMap.NONE
        Case "UPPER"
            iMapType = com.sun.star.style.CaseMap.UPPERCASE
        Case "LOWER"
            iMapType = com.sun.star.style.CaseMap.LOWERCASE
        Case "TITLE"
            iMapType = com.sun.star.style.CaseMap.TITLE
        Case "SMALL CAPS"
            iMapType = com.sun.star.style.CaseMap.SMALLCAPS
        Case Else
            Print "Désolé, " & sMapType & " n'est pas un type reconnu valide "
        End Select
    Loop
    If Not CreateSelectedTextIterator(ThisComponent, _
        "Changer le document entier ?", vCursors()) Then Exit Sub
    For i% = LBound(vCursors()) To UBound(vCursors())
        SetWordCase(vCursors(i%, 0), vCursors(i%, 1), ThisComponent.Text, iMapType%)
    Next
End Sub
'*********************************************************************
'Auteur : Andrew Pitonyak
'email :  andrew@pitonyak.org
Function SetWordCase(vLCursor, vRCursor, vText, iMapType%)
  If IsNull(vLCursor) OR IsNull(vRCursor) OR IsNull(vText) Then Exit Function
```

```
 If vText.compareRegionEnds(vLCursor, vRCursor) <= 0 Then Exit Function
 vLCursor.goRight(0, False)
 Do While vLCursor.gotoNextWord(True)
  If vText.compareRegionStarts(vLCursor, vRCursor) > 0 Then
    vLCursor.charCasemap = iMapType%
    vLCursor.goRight(0, False)
  Else
    Exit Function
  End If
 Loop
 Rem Si le dernier mot du document n'est pas suivi de ponctuation et de nouvelles lignes
 Rem alors il est impossible d'aller au prochain mot. Je dois maintenant m'attaquer à ce cas de figure
 If vText.compareRegionStarts(vLCursor, vRCursor) > 0 AND vLCursor.gotoEndOfWord(True) Then
  vLCursor.charCasemap = iMapType%
 End If
End Function
```

7.15 Andrew apprend à parcourir les paragraphes

Je voulais parcourir les paragraphes et éventuellement changer le style de paragraphe. Pour cela, je devais sélectionner chaque paragraphe tour à tour. Mon premier essai, qui fut une erreur, consistait à démarrer par la sélection d'un paragraphe. Puis je déplaçais le curseur de zéro espace à droite, pour désélectionner le paragraphe courant.

```
vCurCursor.goRight(0, False)
```

Ça marche bien, mais le curseur reste dans le paragraphe courant. Dans OOo, un paragraphe est du genre “blah blah blah.<cr><lf>”. Quand j'utilise la méthode ci-dessus, je laisse le curseur juste avant <cr><lf>, qui font partie du paragraphe. Et quand j'utilisais vCursor.gotoNextParagraph(True), cela sélectionnait <cr><lf> et plaçait le curseur au début du paragraphe suivant . Ce n'est CERTAINEMENT PAS ce que je voulais. La bonne manière de passer au paragraphe suivant et de le sélectionner est la suivante :

```
vCurCursor.gotoNextParagraph(False) 'aller au début du prochain paragraphe.
vCurCursor.gotoEndOfParagraph(True) 'sélectionner le paragraphe.
```

Pour parcourir les paragraphes et imprimer leur style, j'utilise la macro ci-dessous. Attention, si j'ai sélectionné plus d'un paragraphe, je ne peux pas obtenir le style de paragraphe, d'où l'importance de pouvoir sélectionner un seul paragraphe à la fois.

```
Sub PrintAllStyles
 Dim s As String
 Dim vCurCursor as Variant
 Dim vText As Variant
 Dim sCurStyle As String

 vText = ThisComponent.Text
 vCurCursor = vText.CreateTextCursor()
 vCurCursor.GoToStart(False)
 Do
  If NOT vCurCursor.gotoEndOfParagraph(True) Then Exit Do
  sCurStyle = vCurCursor.ParaStyleName
  s = s & """" & sCurStyle & """" & CHR$(10)
 Loop Until NOT vCurCursor.gotoNextParagraph(False)
 MsgBox s, 0, "Styles dans le Document"
End Sub
```

La dernière application consiste à parcourir ce document que vous lisez, en affectant un style de paragraphe aux lignes de code. Dans le cas d'une macro d'une seule ligne de code, le style sera

_code_une_ligne. S'il y a plus d'une ligne, alors la première ligne sera _code_prem_ligne, la dernière sera _code_dern_ligne, et ce qui est entre le deux sera juste _code. Je n'ai pas été très rigoureux sur les conditions de démarrage ou de fin de travail, aussi cela pourra ne pas marcher si j'ai la première ou la dernière ligne en _code (ce qui n'est pas le cas, donc je ne m'inquiète pas pour ça).

```
Sub user_CleanUpCodeSections
  worker_CleanUpCodeSections("_code_prem_ligne", "_code", "_code_dern_ligne", "_code_une_ligne")
End Sub



REM  Je ne peux pas utiliser simplement la commande goRight(0, False) pour avancer le curseur, car je ne veux
REM  pas capturer les caractères <NL>. Beurk !
Sub worker_CleanUpCodeSections(firstStyle$, midStyle$, lastStyle$, onlyStyle$)
  Dim vCurCursor as Variant     ' Curseur actuel
  Dim vPrevCursor as Variant    'Curseur précédent (un paragraphe au-dessus)
  Dim sPrevStyle As String      '  Style précédent
  Dim sCurStyle As String       ' Style actuel

  REM  Place le curseur actuel au début du second paragraphe
  vCurCursor  = ThisComponent.Text.CreateTextCursor()
  vCurCursor.GoToStart(False)
  If NOT vCurCursor.gotoNextParagraph(False) Then Exit Sub

  REM  Place le curseur précédent pour sélectionner le premier paragraphe
  vPrevCursor = ThisComponent.Text.CreateTextCursor()
  vPrevCursor.GoToStart(False)
  If NOT vPrevCursor.gotoEndOfParagraph(True) Then Exit Sub
  sPrevStyle = vPrevCursor.ParaStyleName

  Do
    REM Je ne peux pas utiliser simplement goRight(0,
    If NOT vCurCursor.gotoEndOfParagraph(True) Then Exit Do
    sCurStyle = vCurCursor.ParaStyleName

    REM  C'est là que commence le travail.
    If sCurStyle = firstStyle$ Then
      REM Le style actuel est le premier style
      REM  Je cherche si le style précédent était un de ces styles
      Select Case sPrevStyle
        Case onlyStyle$, lastStyle$
          sCurStyle = midStyle$
          vCurCursor.ParaStyleName = sCurStyle
          vPrevCursor.ParaStyleName = firstStyle$

        Case firstStyle$, midStyle$
          sCurStyle = midStyle$
          vCurCursor.ParaStyleName = sCurStyle
      End Select

    ElseIf sCurStyle = midStyle$ Then
      REM  Le style actuel est le style du milieu
      REM  Idem : je cherche si le style précédent était un de ces styles
      Select Case sPrevStyle
        Case firstStyle$, midStyle$
```

```
      REM  Ne rien faire !

    Case onlyStyle$
      REM Le dernier style était un style seul, mais il arrive avant un style de milieu !
      vPrevCursor.ParaStyleName = firstStyle$

    Case lastStyle$
      vPrevCursor.ParaStyleName = midStyle$

    Case Else
      sCurStyle = firstStyle$
      vCurCursor.ParaStyleName = sCurStyle
  End Select

ElseIf sCurStyle = lastStyle$ Then
  Select Case sPrevStyle
    Case firstStyle$, midStyle$
      REM  Ne rien faire !

    Case onlyStyle$
      REM Le dernier style était un style seul, mais il arrive avant un style de milieu !
      vPrevCursor.ParaStyleName = firstStyle$

    Case lastStyle$
      vPrevCursor.ParaStyleName = midStyle$

    Case Else
      sCurStyle = firstStyle$
      vCurCursor.ParaStyleName = sCurStyle
  End Select

ElseIf sCurStyle = onlyStyle$ Then
  Select Case sPrevStyle
    Case firstStyle$, midStyle$
      sCurStyle = midStyle$
      vCurCursor.ParaStyleName = sCurStyle

    Case lastStyle$
      sCurStyle = lastStyle$
      vCurCursor.ParaStyleName = sCurStyle
      vPrevCursor.ParaStyleName = midStyle$

    Case onlyStyle$
      sCurStyle = lastStyle$
      vCurCursor.ParaStyleName = sCurStyle
      vPrevCursor.ParaStyleName = firstStyle$
  End Select

Else
  Select Case sPrevStyle
    Case firstStyle$
      vPrevCursor.ParaStyleName = onlyStyle$
```

```
      Case midStyle$
        vPrevCursor.ParaStyleName = lastStyle$
      End Select
    End If

    REM  Le travail est fait, donc on avance le curseur
    vPrevCursor.gotoNextParagraph(False)
    vPrevCursor.gotoEndOfParagraph(True)
    sPrevStyle = vPrevCursor.ParaStyleName
  Loop Until NOT vCurCursor.gotoNextParagraph(False)

End Sub
```

7.16 Où est le Curseur affiché ?

Je n'ai pas le temps de détailler, mais voici le mail envoyé par Giuseppe Castagno [castagno@tecsa-srl.it], qui m'a proposé ces idées.

Ce que vous avez écrit est très intéressant, mais je ne suis pas sûr que ce soit correct. D'abord, la position du curseur (commande getposition) semble être relative au premier endroit en haut de la feuille qui puisse contenir du texte. S'il y a un en-tête, la position sera relative au début de cet en-tête, tandis que s'il n'y en a pas, la position sera relative au début de la zone de texte. Il semble aussi que la marge du haut soit comprise entre le haut de la page et le premier endroit qui peut contenir du texte.

Vos mesures de la position du pied de page sont bien imaginées, car cela vous donne l'espace entre le haut du pied de page et le curseur. J'ai trouvé ça génial, je n'y avais jamais pensé. Néanmoins, qu'arrive-t-il si on a augmenté la taille du pied de page ? Je crois que vous n'avez pas pris cela en compte.

Vous pouvez probablement mesurer plutôt comme ceci :

Hauteur de la page – marge du haut – position du curseur

Ainsi, pas besoin de déplacer le curseur.

```
Sub PrintCursorLocation
  Dim xDoc
  Dim xViewCursor
  Dim s As String

  xDoc = ThisComponent
  xViewCursor = xDoc.CurrentController.getViewCursor()
  s = xViewCursor.PageStyleName

  Dim xFamilyNames As Variant, xStyleNames As Variant
  Dim xFamilies
  Dim xStyle, xStyles

  xFamilies = xDoc.StyleFamilies
  xStyles = xFamilies.getByName("PageStyles")
  xStyle  = xStyles.getByName(xViewCursor.PageStyleName)
'  RunSimpleObjectBrowser(xViewCursor)

  Dim lHeight As Long
  Dim lWidth  As Long
  lHeight = xStyle.Height
  lWidth = xStyle.Width
```

```
s = "La taille de la page est  " & CHR$(10) &_
    "     " & CStr(lWidth / 100.0) & " mm par " &_
    "     " & CStr(lHeight / 100.0) & " mm" & CHR$(10) &_
    "     " & CStr(lWidth / 2540.0) & " pouces par " &_
    "     " & CStr(lHeight / 2540.0) & " pouces" & CHR$(10) &_
    "     " & CStr(lWidth *72.0 / 2540.0) & " picas par " &_
    "     " & CStr(lHeight *72.0 / 2540.0) & " picas" & CHR$(10)


Dim dCharHeight As Double
Dim iCurPage As Integer

Dim dXCursor As Double
Dim dYCursor As Double
Dim dXRight  As Double
Dim dYBottom As Double
Dim dBottomMargin As Double
Dim dLeftMargin As Double


dCharHeight = xViewCursor.CharHeight / 72.0
iCurPage =  xViewCursor.getPage()

Dim v
v = xViewCursor.getPosition()
dYCursor = (v.Y + xStyle.TopMargin)/2540.0 + dCharHeight / 2
dXCursor = (v.X + xStyle.LeftMargin)/2540.0
dXRight  = (lWidth - v.X - xStyle.LeftMargin)/2540.0
dYBottom = (lHeight - v.Y - xStyle.TopMargin)/2540.0 - dCharHeight / 2
'  Print "Marge de gauche = " & xStyle.LeftMargin/2540.0


dBottomMargin = xStyle.BottomMargin / 2540.0
dLeftMargin = xStyle.LeftMargin / 2540.0
s = s & "Le curseur est à " & Format(dXCursor, "0.##") & " pouces du bord gauche " & CHR$(10)
s = s & "Le curseur est à  " & Format(dXRight, "0.##") & " pouces du bord droit " & CHR$(10)
s = s & "Le curseur est à  " & Format(dYCursor, "0.##") & " pouces du haut  " & CHR$(10)
s = s & "Le curseur est à  " & Format(dYBottom, "0.##") & " pouces du bas " & CHR$(10)
s = s & "Marge gauche  = " & dLeftMargin & " pouces" & CHR$(10)
s = s & "Marge du bas = " & dBottomMargin & " pouces" & CHR$(10)
s = s & "Hauteur des caractères = " & Format(dCharHeight, "0.####") & " pouces" & CHR$(10)

'  RunSimpleObjectBrowser(xStyle)

'  Dim dFinalX As Double
'  Dim dFinalY As Double
'  dFinalX = dXCursor + dLeftMargin
'  dFinalY = (v.Y + xStyle.TopMargin)/2540 + dCharHeight / 2
'  s = s & "Le curseur dans la page est à (" & Format(dFinalX, "0.####") & ", " &_
'          Format(dFinalY, "0.####") & ") en pouces" & CHR$(10)

  REM  Maintenant vérifions le pied de page !
'  If xStyle.FooterIsOn Then
```

```
'    v = IIF(iCurPage MOD 2 = 0, xStyle.FooterTextLeft, xStyle.FooterTextRight)
'    If IsNull(v) Then v = xStyle.FooterText
'    If Not IsNull(v) Then
'      REM  Sauvergarde la position
'      Dim xOldCursor
'      xOldCursor = xViewCursor.getStart()
'      xViewCursor.gotoRange(v.getStart(), false)
'      Print "Position du pied de page = " & CStr(xViewCursor.getPosition().Y/2540.0)
'      dFinalY = xViewCursor.getPosition().Y/2540.0 - dYCursor + dBottomMargin
'      xViewCursor.gotoRange(xOldCursor, false)
'    End If
'  Else
'    Print "Pas de pied de page"
'  End If
'  s = s & "Le curseur dans la page est à  (" & Format(dFinalX, "0.####") & ", " &_
'          Format(dFinalY, "0.####") & ") en pouces" & CHR$(10)
  MsgBox s, 0, "Information sur la page"
End Sub
```

8Formulaires

Attention	Je connais très peu de choses sur les formulaires. Aussi, les informations données ici peuvent ne pas être tout à fait exactes. Le guide du développeur a de bons exemples sur les formulaires.

Frank Schönheit [fs@openoffice.org] attire l'attention sur ceci :

Pour modifier un contrôle AWT, il faut faire les changements sur le **modèle** et le contrôle qui appartient à ce modèle est automatiquement mis à jour et synchronisé. Changer le contrôle directement pourrait amener à des incohérences. Par exemple, pour une liste de sélection, ne pas utiliser l'interface XlistBox,fournie par le contrôle, pour sélectionner une entrée (XlistBox::selectItem) mais plutôt la propriété com.sun.star.awt.UnoControlListBoxModel::SelectedItems du modèle.

Voir: http://api.openoffice.org/docs/common/ref/com/sun/star/awt/UnoControlListBoxModel.html

8.1Introduction

Un formulaire est un ensemble de 'forms' et/ou de 'contrôles' comme les boutons ou les listes déroulantes. Les formulaires peuvent être utilisés pour accéder à des sources de données ou des choses plus compliquées.

Voir également :
http://api.openoffice.org/docs/common/ref/com/sun/star/form/XFormsSupplier.html
http://api.openoffice.org/docs/common/ref/com/sun/star/form/module-ix.html

Pour obtenir un formulaire en OOBasic, il vous faut d'abord accéder à la 'DrawPage'. La méthode pour l'obtenir dépend du type de document. Comme test, j'ai inséré un bouton dans une feuille de calcul, je l'ai nommé 'TestButton' et le formulaire 'TestForm'. J'ai alors créé la macro suivante qui change l'étiquette du bouton quand on clique dessus.

```
Sub TestButtonClick
Dim vButton, vForm
vForm=THISCOMPONENT.CurrentController.ActiveSheet.DrawPage.Forms.getByName("TestForm")
vButton = vForm.getByName("TestButton")
vButton.Label = "Wow"
Print vButton.getServiceName()
End Sub
```

Si j'avais travaillé avec un document Texte, on obtiendrait le formulaire comme suit :

```
oForm=THISCOMPONENT.DrawPage.Forms.getByName("FormName")
```

8.2Boîtes de dialogue

La plupart des exemples de boîtes de dialogue commencent ainsi :

```
Dim oDlg As Object
Sub StartDialog
    oDlg = CreateUnoDialog(DialogLibraries.Standard.Dialog1 )
    oDlg.execute()
End Sub
```

Notez la variable appelée oDlg qui est définie en dehors de la fonction où elle est créée. Ceci est nécessaire car elle peut être manipulée par d'autres procédures que l'on appelle comme gestionnaires d'événement. Si ces procédures accèdent à la boîte de dialogue, alors elles doivent pouvoir accéder à la variable qui la référence.

Les boîtes de dialogue, comme les bibliothèques de macros, ont leur propre hiérarchie. Dans cet

exemple, j'ai entré la code de la macro montrée ci dessus. Puis j'ai choisi Outils>Macros et j'ai cliqué sur le bouton « Gérer ». Vous noterez que ceci organise les choses sous le document, avec une section nommée Standard. Lorsque vous cliquez sur « nouv. boîte de dialogue », le nom par défaut est « Dialog1 ». Parce que ce code tourne depuis une macro embarquée dans un document, DialogLibraries se réfère à la hiérarchie des librairies du document. Si vous voulez accéder à la hiérarchie des bibliothèques de l'application depuis la macro d'un document, vous devez utiliser GlobalScope.DialogLibraries.

La méthode standard de fermeture d'une boîte de dialogue implique de mettre un gestionnaire d'événement qui la fermera. Généralement, j'ajoute un bouton « Fermer » qui appelle une méthode semblable à ce qui suit :

```
Sub CloseDialog
    oDlg.endExecute()
End Sub
```

Si vous voulez que la boîte de dialogue se ferme d'elle même après une durée spécifiée, vous créez votre routine de fermeture de la boîte de dialogue et vous la reliez à un gestionnaire d'événement approprié.

```
Sub CloseDialogAfterWaiting
    'Attendre deux secondes
    wait(2000)
    oDlg.endExecute()
End Sub
```

S'il vous plaît, excusez la grande quantité de détails présents, mais je sentais que c'était nécessaire pour les novices. Lors de l'édition de ma nouvelle boîte de dialogue, je clique sur le bouton « Champs de contrôle » de la barre d'outils et je choisis un bouton de commande. Je le positionne. Je clique avec le bouton droit sur lui et choisis « propriétés ». A partir de l'onglet « Général », je mets « BoutonQuitter » comme nom pour ce bouton et comme étiquette « Quitter ». Puis j'ouvre l'onglet « Evénements » et à « lors du déclenchement », je clique sur « ... » pour ouvrir la fenêtre qui me permet d'assigner des actions aux événements. Dans l'onglet « Evénements », je choisis « lors du déclenchement ». Dans la partie « Macro », je sélectionne la routine « CloseDialog » et je clique sur « Assigner ». Bien que ceci gère aussi bien l'activation par la souris que par le clavier, il est possible d'avoir une procédure différente pour « KeyPressed » et « MouseClicked » mais je n'ai pas de raisons de les différencier.

Truc	Vous devez avoir une variable référençant la boîte de dialogue en dehors de la routine qui la crée afin de pouvoir y accéder depuis d'autres routines.
Truc	Depuis une macro d'un document, DialogLibraries.Standard.Dialog1 référence une boîte de dialogue dans le document courant. GlobalScope.DialogLibraries.Standard... référence une boîte de dialogue globale.

8.2.1Contrôles

Les contrôles partagent tous certaines fonctionnalités communes. Quelques unes sont ici :

http://api.openoffice.org/docs/common/ref/com/sun/star/awt/UnoControl.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XWindow.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/module-ix.html

Ceci permet de contrôler des choses telles que la visibilité, l'activation, la taille et autres caractéristiques communes. Plusieurs contrôles partagent des méthodes communes telles que setLabel(string). Plusieurs événements différents sont supportés. Par expérience, les plus communément utilisés se rapportent aux notifications de changements d'état des contrôles.

Un contrôle peut être obtenu depuis une boîte de dialogue utilisant la méthode getControl (control_name). Il est également possible de faire des itérations au travers de l'ensemble des contrôles si vous le désirez.

8.2.2Champs d'étiquette (label)

Un champ d'étiquette agit comme un texte régulier dans la boîte de dialogue. Il est habituellement utilisé pour étiqueter un contrôle. Il est possible de spécifier ou de récupérer le texte du champ d'étiquette en utilisant getText() et setText(string). Il est également possible de préciser l'alignement du texte à gauche, centré, ou à droite. Il est courant d'aligner à droite le texte d'un champs d'étiquette afin qu'il soit au plus près du contrôle qu'il identifie. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XFixedText.html

8.2.3Bouton

Généralement, un contrôle Bouton est seulement utilisé pour appeler une procédure lorsque le bouton est pressé. Il est également possible d'appeler setLabel(string) pour changer le texte du bouton. Voir aussi :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XButton.html

8.2.4Zone de texte

Une zone de texte est utilisée pour saisir du texte standard. Il est possible de limiter la taille maximum du texte et de contrôler le nombre maximum de lignes. Il est possible d'écrire vos propres systèmes de formatage si vous le souhaitez. Les méthodes les plus utilisées sont getText() et setText(string). Si vous voulez un ascenseur, vous devez le paramétrer dans les propriétés de la zone de Texte pendant la conception de la boîte de dialogue.

Il y a des boîtes de saisie particulières pour les dates, heures, nombres, masques de saisie, champs formatés et les monnaies. Si vous en utilisez, soyez certain de prêter une attention particulière aux propriétés pour voir ce qu'elles peuvent faire. Vous pouvez désactiver un contrôle de format strict ou fournir des plages d'entrées limitées, par exemple. Un champ masqué dispose d'un masque de saisie et d'un masque de caractère. Le masque de saisie détermine les données que l'utilisateur peut entrer. Le masque de caractères détermine l'état du champ masqué lors du chargement du formulaire. Le champ formaté autorise un formatage arbitraire permis par OOo. Si je voulais un champ pour un numéro de sécurité sociale ou pour un pourcentage, j'utiliserais ce champ.

Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XTextComponent.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XTimeField.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XDateField.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XCurrencyField.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XNumericField.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XPatternField.html

8.2.5Zone de liste

Une zone de liste fournit une liste de valeurs parmi lesquelles vous pouvez en sélectionner une. Vous pouvez choisir d'activer la sélection multiple. Pour additionner des éléments à la liste, j'utilise couramment quelque chose du type addItems(Array("one", "two", "three"), 0). Il est également possible d'ôter des éléments d'une zone de liste.

Pour une sélection unique, vous pouvez utiliser getSelectedItemPos() pour déterminer quel élément est sélectionné. -1 est renvoyé si il n'y a rien de sélectionné. Si quelque chose est sélectionnée, 0 signifie que c'est le premier de la liste. Pour des sélections multiples, utilisez getSelectedItemsPos() qui retourne un tableau contenant les indexes sélectionnés de la liste. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XListBox.html

8.2.6Zone combinée

Une zone combinée est un champ d'entrée avec une zone de liste liée. Ceci est parfois appelé une liste déroulante. Dans mon exemple, j'ai placé une zone combinée en deux temps. D'abord j'entre les valeurs de la liste et je positionne la zone combinée sur propriété :

```
aControl.addItems(Array("properties", "methods", "services"), 0)
aControl.setText("properties")
```

Je vais ensuite dans les événements et j'indique pour “statut changé” que la routine NewDebugType peut être appelée. Ceci affichera l'intégralité des méthodes, propriétés et services supportés par la boîte de dialogue. J'ai fait ceci pour montrer qu'un appel en retour d'un événement est réellement utile pour obtenir des informations de débogage. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XComboBox.html

8.2.7Case à cocher

Bien qu'une case à cocher soit généralement utilisée pour indiquer un unique état (« oui » ou « non »), il existe une case à cocher commune nommée case à cocher à trois état. Dans OOo une case à cocher peut avoir l'état 0, non cochée, ou 1, cochée. Vous pouvez utiliser getState() pour obtenir l'état courant. Si vous appelez enableTriState(true), alors l'état 2 est autorisé. Cet état positionne une valeur grisée dans la case traduisant un état intermédiaire 'au tout ou rien habituel'. Vous pouvez paramétrer l'état en utilisant setState(int). Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XCheckBox.html

8.2.8Bouton d'option/Radio

L'utilité d'un bouton radio est généralement de sélectionner un choix parmi plusieurs propositions. Pour cette raison, on insère habituellement, d'abord une zone de groupe puis on place les boutons radio dedans. Pour savoir lequel a été sélectionné, vous pouvez appeler la méthode getState() sur chacun d'eux jusqu'à ce que vous le trouviez. J'ai pu utiliser les boutons radio plutôt qu'une liste déroulante dans mon exemple pour choisir quoi afficher dans ma boîte de liste de débogage. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XRadioButton

8.2.9Barre de progression

Dans mon exemple, j'utilise une barre de progression pour montrer la progression du remplissage d'une boîte de liste de débogage. Ce n'est probablement pas un bon exemple parce que cela se produit trop vite mais cela reste un exemple.

Vous pouvez définir la plage de progression avec setRange(min, max). Ceci rend plus facile la mise à jour de la progression. Comme je traite une chaîne de caractère, je mets le minimum à 0 et le maximum à la longueur de la chaîne. J'appelle alors setValue(int) avec la position courante dans la chaîne pour montrer où j'en suis. Voir :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XProgressBar.html

8.3Obtention des Contrôles

Si vous voulez énumérer les contrôles présents dans un formulaire, ce qui suit marchera :

```
Sub EnumerateControlsInForm
    Dim oForm, oControl, iNumControls%, i%
    'Par défaut, c'est là que se trouvent lescontrôles
    oForm = ThisComponent.Drawpage.Forms.getByName("Standard")
    oControl = oForm.getByName("MyPushButton")
    MsgBox "getByName est utilisé pour obtenir le contrôle appelé " & oControl.Name
    iNumControls = oForm.Count()
    For i = 0 To iNumControls -1
```

```
        MsgBox "Le contrôle " & i & " est nommé " & oControl.Name
    Next
End Sub
```

Pour les contrôles dans une boîte de dialogue, utiliser le code qui suit :

```
    x = oDlg.getControls()
    For ii=LBound(x) To UBound(x)
        Print x(ii).getImplementationName()
    Next
```

Généralement, les contrôles seront recherchés par noms plutôt que par énumérations.

8.3.1Informations sur un contrôle

La solution a finalement été fournie par Paolo Mantovani [mantovani.paolo@tin.it]

Un contrôle de formulaire est placé sur une forme de dessin (com.sun.star.drawing.XControlShape), il faut donc obtenir la forme sous-jacente afin de gérer la taille et la position du contrôle.

La bibliothèque Outils (module "Module Controls") fournit quelques méthodes qui permettent de travailler avec des contrôles de formulaire, notamment les fonctions suivantes :

```
GetControlModel()
GetControlShape()
GetControlView()
```

En espérant que cela vous aide... Paolo Mantovani

En effet Paolo, cela m'a permis d'écrire la macro suivante :

```
Sub SeeThingProp
    Dim oForm, oControl, iNumControls%, i%
    Dim v()
    'Par défaut, les contrôles se trouvent ici
    oForm = ThisComponent.Drawpage.Forms.getByName("Standard")
'   oControl = oForm.getByName("MyPushButton")
    oControl = oForm.getByName("CheckBox")
    Dim vShape
    Dim vPosition, vSize
    Dim s$, vv
    vShape = GetControlShape(ThisComponent, "CheckBox")
    vPosition = vShape.getPosition()
    vSize = vShape.getSize()
'   RunSimpleObjectBrowser(vSize)
    s = s & "Position = (" & vPosition.X & ", " & vPosition.Y & ")" & CHR$(10)
    s = s & "Height = " & vSize.Height & " Width = " & vSize.Width & CHR$(10)
    MsgBox s
End Sub
```

8.4Sélection d'un fichier depuis une boîte de dialogue Fichier

L'exemple suivant affiche la boîte standard de dialogue Fichier

```
Sub ExampleGetAFileName
    Dim filterNames(1) As String
    filterNames(0) = "*.txt"
    filterNames(1) = "*.sxw"
    Print GetAFileName(filterNames())
End Sub

Function GetAFileName(Filternames()) As String
    Dim oFileDialog as Object
```

```
    Dim iAccept as Integer
    Dim sPath as String
    Dim InitPath as String
    Dim RefControlName as String
    Dim oUcb as object
    'Dim ListAny(0)
    'Nota : Les services suivants doivent être appelés dans l'ordre suivant
    ' sinon le Basic n'enlève pas le service FileDialog
    oFileDialog = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")
    oUcb = createUnoService("com.sun.star.ucb.SimpleFileAccess")
    'ListAny(0) = com.sun.star.ui.dialogs.TemplateDescription.FILEOPEN_SIMPLE
    'oFileDialog.initialize(ListAny())
    AddFiltersToDialog(FilterNames(), oFileDialog)

    'Mets ton chemin initial ici !
    'InitPath = ConvertToUrl(oRefModel.Text)

    If InitPath = "" Then
        InitPath = GetPathSettings("Work")
    End If
    If oUcb.Exists(InitPath) Then
        oFileDialog.SetDisplayDirectory(InitPath)
    End If
    iAccept = oFileDialog.Execute()
    If iAccept = 1 Then
        sPath = oFileDialog.Files(0)
        GetAFileName = sPath
        'If oUcb.Exists(sPath) Then
        '    oRefModel.Text = ConvertFromUrl(sPath)
        'End If
    End If
    oFileDialog.Dispose()
End Function
```

8.5Centrage d'une boîte de dialogue à l'écran

Remerciements à Berend Cornelius [Berend.Cornelius@sun.com] d'avoir fourni cette macro. L'astuce réside dans l'utilisation du contrôleur courant afin de récupérer la fenêtre du composant et, à partir de là, récupérer la position et la taille de la fenêtre.

```
Sub Main
    Dim CurPosSize as new com.sun.star.awt.Rectangle
    FramePosSize = ThisComponent.getCurrentController().Frame.getComponentWindow.PosSize
    xWindowPeer = oDialog.getPeer()
    CurPosSize = oDialog.getPosSize()
    WindowHeight = FramePosSize.Height
    WindowWidth = FramePosSize.Width
    DialogWidth = CurPosSize.Width
    DialogHeight = CurPosSize.Height
    iXPos = ((WindowWidth/2) - (DialogWidth/2))
    iYPos = ((WindowHeight/2) - (DialogHeight/2))
    oDialog.setPosSize(iXPos, iYPos, DialogWidth, DialogHeight, com.sun.star.awt.PosSize.POS)
    oDialog.execute()
End Sub
```

8.6Programmer les réactions aux évènements de contrôle

Le petit extrait de code suivant assigne les évènements d'une boîte de dialogue à la macro Test du module Module1 de la librairie standard. Merci à Oliver Brinzing [OliverBrinzing@t-online.de] pour ce code.

```
Sub SetEvent
    Dim oDocument as Object
    Dim oView as Object
    Dim oDrawPage as Object
    Dim oForm as Object
    Dim oEvents(0) as New com.sun.star.script.ScriptEventDescriptor

    oDocument = StarDesktop.getCurrentComponent
    oView = oDocument.CurrentController
    oDrawPage = oView.getActiveSheet.DrawPage

'   atteindre le premier formulaire
    oForm = oDrawPage.getForms.getByIndex(0)

    oEvents(0).ListenerType = "XActionListener"
    oEvents(0).EventMethod  = "actionPerformed"
    oEvents(0).AddListenerParam = ""
    oEvents(0).ScriptType = "StarBasic"

    oEvents(0).ScriptCode = "document:Standard.Module1.Test"
    oForm.registerScriptEvent(0, oEvents(0))
End Sub


Sub Test(oEvt)
    Print oEvt.Source.Model.Name
End Sub
```

Je suppose qu'une brève description sur la macro registrerScriptEvent est nécessaire. Regardez http://api.openoffice.org/docs/common/ref/com/sun/star/script/XEventAttacherManager.html pour de bons éléments de départ.

```
registerScriptEvent(index, ScriptEventDescriptor)
```

La variable oEvents est un tableau de descripteurs d'évènements, voir la page http://api.openoffice.org/docs/common/ref/com/sun/star/script/ScriptEventDescriptor.html, qui décrit l'évènement qui sera attaché (EventMethod) et quelle macro sera appelée (ScriptCode). L'exemple ci-dessus aurait pu utiliser une simple variable au lieu d'un tableau parce que le code utilise une seule entrée du tableau. Le premier argument passé à la méthode registrerScriptEvent est un index vers l'objet qui « utilisera » le descripteur d'évènements. Les pages d'aide considèrent que vous connaissez quels objets sont indexés et indiquent que « si un objet est attaché à cet index, alors l'événement sera attaché automatiquement ». Ce qui n'est pas précisé est que le formulaire agit comme un conteneur d'objets pour les composants de type formulaire. Ces composants de type formulaire sont accessibles par leur nom ou leur index.

8.7Comment rendre une boîte de dialogue non-modale

CP Hennessy [cphennessy@openoffice.org] a posé cette question et l'a résolue avec Python.

Comment rendre une boîte de dialogue UnoControlDialog non-modale ?

```
'Attention Ceci est une syntaxe Python
```

```
    toolkit = smgr.createInstanceWithContext( "com.sun.star.awt.Toolkit", ctx)
    aRect = uno.createUnoStruct( "com.sun.star.awt.Rectangle" )
    aRect.X = 100
    aRect.Y = 100
    aRect.Width = posX
    aRect.Height = 25
    aDescriptor = uno.createUnoStruct( "com.sun.star.awt.WindowDescriptor");
    aDescriptor.Type =   TOP;
    aDescriptor.WindowServiceName   =   "workwindow";
    aDescriptor.ParentIndex        =   1;
    aDescriptor.Bounds             =   aRect;
    peer = toolkit.createWindow( aDescriptor )

  self.dialog.createPeer( toolkit, peer )
```

Je (Andrew Pitonyak) n'ai pas testé cette méthode, ni même essayé de la convertir en OOo Basic, mais cela devrait fonctionner.

8.8Interception des Entrées Clavier

Une question a été posée pour lier l'évènement “touche enfoncée” dans un champ de texte à une sous-routine. Comment savoir quelle était la touche enfoncée ? Laurent Godard a fourni la réponse suivante :

```
Sub MySub(oEvent as object)
  If oEvent.keycode=com.sun.star.awt.key.RETURN Then
    MsgBox "RETURN Pressed"
  End If
End Sub
```

8.9Création d'une boîte de dialogue par programmation

Ceci représente en réalité deux problèmes. Il faut pouvoir créer une boîte de dialogue et puis y ajouter des contrôles. Ces deux situations ont été traitées par Paolo Mantovani et Thomas Benisch.

8.9.1Insertion d'un Contrôle dans une Boîte de Dialogue Existante

Beaucoup de questions trouvent réponse dans le guide du développeur, notamment dans le chapitre 'Basic and Dialogs', sections 'Programming Dialogs and Dialog Controls' et 'Creating Dialogs at Runtime'. Le code suivant insère un contrôle dans une boîte de dialogue existante.

```
Sub Main
    Dim oDialog As Object
    Dim oDialogModel As Object
    Dim oButtonModel As Object

    Const sButtonName = "Button1"

    REM charger la bibliothèque d'outils
    BasicLibraries.LoadLibrary( "Tools" )

    REM charger la boîte de dialogue
    oDialog = LoadDialog( "Standard", "Dialog1" )

    REM obtenir le modèle de la boîte de dialogue
    oDialogModel = oDialog.getModel()

    REM créer un modèle de bouton
```

```
    oButtonModel = oDialogModel.createInstance( _
    "com.sun.star.awt.UnoControlButtonModel" )
    oButtonModel.PositionX = 140
    oButtonModel.PositionY = 120
    oButtonModel.Width = 50
    oButtonModel.Height = 20
    oButtonModel.Label = "OK"
    oButtonModel.Name = sButtonName
    oButtonModel.TabIndex = 1

    REM insérer le modèle de bouton dans le modèle de boîte de dialogue
    oDialogModel.insertByName( sButtonName, oButtonModel )
    REM afficher la boîte de dialogue
    oDialog.execute()
End Sub
```

8.9.2Création d'une Boîte de Dialogue

Un exemple conséquent est fourni dans cette section. Le code pour la création de la boîte de dialogue est compliqué parce que je ne maîtrise pas assez les concepts subtils de l'awt, mais il fonctionne. (Note d'Andrew : les variables ne sont pas déclarées, donc le code échouera si on utilise Option Explicit)

```
REM  *****  BASIC  *****
Global oDlg As Object
Global oButton As Object
Global oTextEdit As Object

' Création d'une boîte de dialogue StarBasic à partir de rien et ajouter quelques contrôles.
Sub ExampleDialogFromScratch

' Obtenir la boîte de dialogue
    sTitle = "Créée à partir de rien!!"
    aRect =  CreateUnoStruct("com.sun.star.awt.Rectangle")
        aRect.Y = 150
        aRect.X = 150
        aRect.Width = 400
        aRect.Height = 250

    oDlg = CreateUnoDialogFromScratch(sTitle, aRect)

' Ajouter un bouton de Commande
    sControlType = "Button"
    sControlName = "BoutonCommande1"
    aRect =  CreateUnoStruct("com.sun.star.awt.Rectangle")
        aRect.Y = 5
        aRect.X = 5
        aRect.Width = 50
  aRect.Height = 12

    oButton = AddControl(oDlg, sControlType, sControlName, aRect)

    oButton.Label = "Cliquer ici..."

' Ajouter un Champ de Texte Editable
```

```
    sControlType = "Edit"
    sControlName = "TexteEdit1"
    aRect =  CreateUnoStruct("com.sun.star.awt.Rectangle")
        aRect.Y = 5
        aRect.X = 60
        aRect.Width = 100
        aRect.Height = 12

    oTextEdit = AddControl(oDlg, sControlType, sControlName, aRect)

    oTextEdit.Text = "Salutation de France !!! :-)"

' Gestion des évènements
    oActionListener = _
    createUnoListener("oButton_", "com.sun.star.awt.XActionListener")

    oButton.addActionListener (oActionListener)

'Affichage de la boîte de dialogue
    oDlg.execute

' Nettoyage des objets
    oButton.removeActionListener (oActionListener)
    oDlg.dispose
End Sub

'*****************************************************************
' Gestion des Evènements
'*****************************************************************
Sub oButton_disposing(oEvt)
'    rien à faire
End Sub

Sub oButton_actionPerformed(oEvt)
    nColor = cLng(Rnd * 255 * 255 * 255)
    oTextEdit.Model.BackGroundColor = nColor
End Sub

'*****************************************************************
' Fonctions d'Assistance
'*****************************************************************
Function AddControl(oDialog, sControlType, sControlName, aPosSize) As Object
    oDlgModel = oDialog.getModel

    sUnoName = "com.sun.star.awt.UnoControl" & sControlType & "Model"
    oControlModel = oDlgModel.createInstance( sUnoName )

    With oControlModel
        .PositionX = aPosSize.X
        .PositionY = aPosSize.Y
        .Width = aPosSize.Width
        .Height = aPosSize.Height
        .Enabled = True
```

```
    End With

    ' Insertion du modèle de contrôle dans le modèle de boîte de dialogue
    oDlgModel.insertByName(sControlName, oControlModel)

    ' ce qui donne la vue du contrôle
    AddControl = oDialog.getControl(sControlName)
End Function


Function CreateUnoDialogFromScratch( sTitle, aPosSize) As Object
    oDlgView = createUnoService("com.sun.star.awt.UnoControlDialog")
    oDlgModel = createUnoService("com.sun.star.awt.UnoControlDialogModel")

    oDlgView.setModel(oDlgModel)

' ceci semble parfois être nécessaire afin d'éviter des erreurs
' mais les valeurs ne semblent avoir aucun effet
'    dim aRect as new com.sun.star.awt.Rectangle
'    aRect.Y    = 0
'    aRect.X   = 0
'    aRect.width  = 0
'    aRect.height = 0

    dim aDescriptor as new com.sun.star.awt.WindowDescriptor

    ' autres valeurs : TOP, MODALTOP, CONTAINER,
    ' je ne connais rien de leurs effets
    aDescriptor.Type = com.sun.star.awt.WindowClass.SIMPLE

'    aDescriptor.WindowServiceName = "testdialog"
'    aDescriptor.ParentIndex = 0
'    aDescriptor.Parent = Null
'    aDescriptor.Bounds = aRect
'    est-ce vraiment nécessaire?

    oToolkit = createUnoService("com.sun.star.awt.Toolkit")
    oWindow = oToolkit.createWindow(aDescriptor)

    oDlgView.createPeer(oToolkit, oWindow)

    oDlgView.setPosSize( _
                aPosSize.X, _
                aPosSize.Y, _
                aPosSize.Width, _
                aPosSize.Height, _
                com.sun.star.awt.PosSize.POSSIZE)

    oDlgView.setTitle(sTitle)

    CreateUnoDialogFromScratch = oDlgView
End Function
```

9Exemple de gestion de placements

NdT : En accord avec l'auteur, les éléments contenus dans ce chapitre ont été fusionnés avec le chapitre 5

10Gestionnaires (handlers) et auditeurs (listeners) d'événements

Un gestionnaire, au sens de ce chapitre, est tout code qui utilise des appels en retour ou qui est amené d'une manière ou d'une autre à traiter certains événements.

10.1 XKeyHandler

Cet exemple de gestionnaire est fourni par Leston Buell [bulbul@ucla.edu]. Sa description du code suit :

10.1.1 Description de la macro Compose réduite

Appeler “Compose” et taper une combinaison de deux lettres choisie parmi ce qui suit :

Ch, ch, Gh, gh, Hh, hh, Jh, jh, Sh, sh, Uh, uh

Celles-ci insèrent respectivement les caractères d'espéranto suivants dans le document :

Ĉ, ĉ, Ĝ, ĝ, Ĥ, ĥ, Ĵ, ĵ, Ŝ, ŝ, Ŭ, ŭ

Note : peu de fontes ont ces caractères.

10.1.2 Commentaires de Leston

Commentaires de Leston avec seulement une modification mineure : Je suis un développeur Java/Python/Javascript et je ne connais pas grand chose au Basic. Ceci pourrait être réécrit pour être modulaire et extensible. Par exemple, on devrait pouvoir charger différents jeux de caractères comme un jeu latin, un jeu hébreu, un jeu cyrillique et les symboles mathématiques. Ces jeux pourraient être dans un tableau plutôt que de faire appel au long "Select...case" que j'ai utilisé. Ceci vous permettrait de générer une table des jeux à la volée. On pourrait également modifier ce code pour travailler sur plus d'un document.

Ceci dit, cette macro fonctionne et satisfait à mes besoins actuels. Ce document contient une version épurée, qui dispose seulement des caractères nécessaires pour l'espéranto. La vraie version, qu'à un certain point je rendrai disponible sur le web, supporte toutes les lettres non-ASCII de Latin1, ExtendedLatinA et ExtendedLatinB, plus d'autres particularités comme les guillemets français et le symbole de l'euro.

Pour utiliser cette macro, appeler "Compose" et saisir une des combinaisons de caractères : "Gh", "gh", "^J", "uh" – ou n'importe quelle autre montrée dans la fonction GetTranslation. Le caractère Unicode correspondant est inséré dans le document. KeyHandler se désenregistre après deux tours, qu'il y ait eu traduction ou non.

J'ai (Leston) assigné Compose à Ctrl+H, ce qui est facile à taper (Note d'*Andrew : Vous pourrez également l'affecter à d'autres touches pour supporter d'autres raccourcis tels que les touches de contrôle de Word Star).*

Je ne comprends pas les portées des variables du Basic, aussi j'ai tout mis en variables globales au début. Si cela n'était pas la bonne manière de procéder, sentez vous libre de changer cela (Note d'*Andrew : Ceci peut amener des problèmes inattendus*). Je pense qu'il y a un bug dans la macro. Ce bug est noté en commentaire dans la fonction InsertString.

10.1.3 Implémentation

Lorsque que la version finale sera disponible, je l'hébergerai moi-même ou je fournirai un lien.

```
REM Auteur:Leston Buell [bulbul@ucla.edu]
Global oComposerDocView
Global oComposerKeyHandler
Global oComposerInputString
```

```
Sub Compose
  oComposerDocView = ThisComponent.getCurrentController
  oComposerKeyHandler = createUnoListener( "Composer_", _
    "com.sun.star.awt.XKeyHandler" )
  oComposerDocView.addKeyHandler( oComposerKeyHandler )
  oComposerInputString = ""
End Sub

Sub ExitCompose
  oComposerDocView.removeKeyHandler( oComposerKeyHandler )
  oComposerInputString = ""
End Sub

Function InsertString( oString, oDocView, oKeyHandler )
  Dim oViewCursor
  Dim oText
  Dim oCursor

  oViewCursor = ThisComponent.getCurrentController().getViewCursor()
  oText = ThisComponent.getText()
  oCursor = oText.createTextCursorByRange( oViewCursor.getStart() )
  'Pour une raison indéterminée, l'insertion de texte relance les événements des touches (deux fois !)
  'Aussi nous enlevons le gestionnaire avant l'insertion, puis le remettons après
  oDocView.removeKeyHandler( oKeyHandler )
  oText.insertString( oCursor.getStart(), oString, true )
  oDocView.addKeyHandler( oKeyHandler )
End Function

Function Composer_keyPressed( oEvt ) as Boolean
  If len( oComposerInputString ) = 1 Then
    oComposerInputString = oComposerInputString & oEvt.KeyChar
    Dim translation
    translation = GetTranslation( oComposerInputString )
    InsertString( translation, oComposerDocView, oComposerKeyHandler )
    oComposerInputString = ""
    ExitCompose
  Else
    oComposerInputString = oComposerInputString & oEvt.KeyChar
  End If
  Composer_KeyPressed = True
End Function

Function Composer_keyReleased( oEvt ) as Boolean
  Composer_keyReleased = False
End Function

Function GetTranslation( oString ) as String
  Select Case oString
    Case "^C", "Ch"
      GetTranslation = "Ĉ"
    Case "^c", "ch"
      GetTranslation = "ĉ"
```

```
  Case "^G", "Gh"
    GetTranslation = "Ĝ"
  Case "^g", "gh"
    GetTranslation = "ĝ"
  Case "^H", "Hh"
    GetTranslation = "Ĥ"
  Case "^h", "hh"
    GetTranslation = "ĥ"
  Case "^J", "Jh"
    GetTranslation = "Ĵ"
  Case "^j", "jh"
    GetTranslation = "ĵ"
  Case "^S", "Sh"
    GetTranslation = "Ŝ"
  Case "^s", "sh"
    GetTranslation = "ŝ"
  Case "uU", "Uh"
    GetTranslation = "Ŭ"
  Case "uu", "uh"
    GetTranslation = "ŭ"
 End Select
End Function
```

10.2 Description des Auditeurs d'événements par Paolo Mantovani

Le texte dans cette nouvelle section a été écrit par Paolo Mantovani. J'ai (Andrew Pitonyak) effectué quelques modifications mineures mais je n'ai pas tenté de vérifier ce code. Ceci dit, je peux vous assurer que Paolo est une source extrêmement fiable. Merci Paolo d'avoir pris le temps. C'est l'une des meilleures descriptions que j'ai vues sur le sujet. Le document contenait ces mentions légales quand je l'ai reçu, aussi je les inclus ici :

10.2.1 La fonction CreateUnoListener

L'environnement d'exécution de OOBasic fournit une fonction nommée CreateUnoListener. Cette fonction comprend deux arguments : un préfixe (String) et un nom pleinement qualifié d'interface d'auditeur d'événement (String). Voici la syntaxe :

```
oListener = CreateUnoListener( sPrefixe , sInterfaceName )
```

Depuis les versions 1.1, cette fonction est très bien décrite dans l'aide du Basic OO, mais nous voudrions ajouter quelques considérations à son sujet. Considérez le code suivant :

```
oListener = CreateUnoListener("prefix_", "com.sun.star.lang.XEventListener")
MsgBox oListener.Dbg_supportedInterfaces
MsgBox oListener.Dbg_methods
```

L'interface com.sun.star.lang.XEventListener est à la base des autres auditeurs d'événements, toutefois c'est le plus simple que vous puissiez trouver. Notez que cette interface travaille comme interface de base pour les autres, donc vous ne devriez pas l'utiliser de manière explicite, mais ce code marche très bien pour cet exemple.

10.2.2 Joli, mais qu'est ce qu'il fait ?

Les macros Basic sont capables d'appeler les méthodes et les propriétés de l'API. Généralement, une macro fait beaucoup d'appels à l'API. A l'inverse, l'API ne peut généralement pas appeler des routines Basic. Considérez l'exemple suivant :

```
Sub Example_Listener
  oListener = CreateUnoListener("prefix_", "com.sun.star.lang.XEventListener")
  Dim oArg As New com.sun.star.lang.EventObject
  oListener.disposing( oArgument )
End Sub

Sub prefix_disposing( vArgument )
   MsgBox "Hi all!!"
End Sub
```

Lorsque la première routine appelle “oListener.disposing()”, la routine prefix_disposing est appelée. Toutefois, la fonction CreateUnoListener crée un service capable d'appeler les procédures OOBasic.

Vous devrez créer les routines et sous-routines avec des noms correspondants à ceux des auditeurs d'événement en ajoutant le préfixe spécifié lorsque que vous appelez CreateUnoListener. Par exemple, l'appel de CreateUnoListener passe un premier argument “prefix_” et la sous-routine “prefix_disposing” démarre avec “prefix_”.

La documentation pour l'interface com.sun.star.lang.XEventListener dit que l'argument doit être une structure com.sun.star.lang.EventObject.

10.2.3 Tout à fait inutile, mais dites-m'en davantage...

Bien que ceci ne soit pas le meilleur moyen pour appeler une macro depuis une autre, UNO nécessite un auditeur d'événement pour appeler vos macros. Lorsque vous voudrez utiliser un auditeur pour intercepter des événements, vous aurez besoin d'un objet capable de parler à votre auditeur. L'objet UNO qui appellera votre auditeur est nommé un "broadcaster". Les objets UNO broadcaster utilisent des méthodes pour ajouter ou ôter les auditeurs appropriés. Pour créer un objet auditeur, passez le nom pleinement qualifié de l'interface de l'auditeur à la fonction CreateUnoListener. Récupérez les méthodes supportées par l'auditeur en accédant aux propriétés de la méthode Dbg_methods property (ou vérifiez la documentation IDL pour les interfaces d'auditeurs). Finalement, implémentez une routine basic pour chaque méthode, même pour celle qui libère l'objet.

La plupart des services UNO fournissent des méthodes pour enregistrer et déréférencer les auditeurs. Par exemple, le service com.sun.star.OfficeDocumentView supporte l'interface com.sun.star.view.XSelectionSupplier. Ceci est le broadcaster. Cette interface fournit les méthodes suivantes :

```
addSelectionChangeListener
```

enregistre un auditeur d'événement, qui est appelé lorsque la sélection change.

```
removeSelectionChangeListener
```

déréférence l'auditeur qui a été enregistré par addSelectionChangeListener.

Les deux méthodes prennent un argument SelectionChangeListener (qui est un service qui supportant l'interface com.sun.star.view.XSelectionChangeListener).

L'objet broadcaster introduit un ou plusieurs arguments dans l'appel. Le premier argument est une structure UNO, les suivants dépendent de la définition de l'interface. Vérifiez la documentation IDL de l'interface de l'auditeur que vous utilisez et voyez le détail des méthodes. Souvent, la structure passée est un objet com.sun.star.lang.EventObject. Toutefois, toutes les structures d'événement doivent étendre com.sun.star.lang.EventObject donc elles doivent avoir au moins l'élément source.

10.2.4 Exemple 1 : com.sun.star.view.XSelectionChangeListener

Ce qui suit est une implémentation complète d'un auditeur de changement de sélection. Il peut être

utilisé avec tous les documents OpenOffice.org.

```
Option Explicit

Global oListener As Object
Global oDocView As Object

'exécuter cette macro démarre l'interception d'événement
Sub Example_SelectionChangeListener
  oDocView = ThisComponent.getCurrentController

  'Crée un auditeur pour intercepter l'événement changement de sélection
  oListener = CreateUnoListener( "MyApp_", "com.sun.star.view.XSelectionChangeListener" )

  ' enregistre l'auditeur auprès du contrôleur du document
  oDocView.addSelectionChangeListener(oListener)
End Sub

'exécuter cette macro stoppe l'interception d'événement
Sub Remove_Listener
  ' déréférence l'auditeur
  oDocView.removeSelectionChangeListener(oListener)
End Sub

'Tous les auditeurs doivent supporter cet événement
Sub MyApp_disposing(oEvent)
  msgbox "disposing the listener"
End Sub

Sub MyApp_selectionChanged(oEvent)
  Dim oCurrentSelection As Object
  'La propriété source de la structure d'événement
  'prend une référence dans la sélection courante
   oCurrentSelection =  oEvent.source
   MsgBox  oCurrentSelection.dbg_properties
End sub
```

Notez que toutes les méthodes d'auditeurs doivent être implémentées dans votre programme basic car si le service appelant ne trouve pas la routine appropriée, une erreur d'exécution se produit.

Références de l' API concernées :
http://api.openoffice.org/docs/common/ref/com/sun/star/view/OfficeDocumentView.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XSelectionSupplier.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XSelectionChangeListener.html
http://api.openoffice.org/docs/common/ref/com/sun/star/lang/EventObject.html

10.2.5 Exemple 2 : com.sun.star.view.XPrintJobListener

Un objet imprimable (disons un objet document) peut supporter l'interface com.sun.star.view.XPrintJobBroadcaster. Cette interface vous autorise à enregistrer (et à déréférencer) un com.sun.star.view.XPrintJobListener pour intercepter les événements lors de l'impression. Lorsque vous en interceptez, vous obtenez une structure com.sun.star.view.PrintJobEvent structure. Cette structure a généralement une propriété “source” ; la source de cet évenement est un Print Job, qui est le service décrivant le travail courant d'impression et qui doit supporter l'interface com.sun.star.view.XPrintJob.

```
Option Explicit
```

```
Global oPrintJobListener As Object

'exécuter cette macro démarre l'interception d'événement
Sub Register_PrintJobListener

  oPrintJobListener = _
  CreateUnoListener("MyApp_", "com.sun.star.view.XPrintJobListener")

  'Cette fonction est définie dans la bibliothèque "Tools"
  'writedbginfo oPrintJobListener

  ThisComponent.addPrintJobListener(oPrintJobListener)

End Sub

'exécuter cette macro stoppe l'interception d'événement
Sub Unregister_PrintJobListener

  ThisComponent.removePrintJobListener(oPrintJobListener)
End Sub

'Tous les auditeurs doivent supporter cet événement
Sub MyApp_disposing(oEvent)
  'Rien à faire ici
End sub

'Cet événement est appelé plusieurs fois
'durant l'impression
Sub MyApp_printJobEvent(oEvent)

  'la source de l'événement printJob est un PrintJob,
  'qui est un service supportant l'interface com.sun.star.view.XPrintJob
  'qui est le service décrivant le travail courant d'impression.
  MsgBox oEvent.source.Dbg_methods

  Select Case oEvent.State

    Case com.sun.star.view.PrintableState.JOB_STARTED
      Msgbox "'L'impression (mise en forme du document) a démarré"

    Case com.sun.star.view.PrintableState.JOB_COMPLETED
       sMsg = "L'impression (mise en forme du document)  "
       sMsg = sMsg & "est terminée, le spoulage a démarré"
      Msgbox sMsg

    Case com.sun.star.view.PrintableState.JOB_SPOOLED
      sMsg = "Spoulage terminé avec succès."
      sMsg = sMsg & " C'est le seul état qui "
      sMsg = sMsg & "puisse être considéré comme un 'succès'"
      sMsg = sMsg & "pour un travail d'impression"
      Msgbox sMsg
```

```
  Case com.sun.star.view.PrintableState.JOB_ABORTED
    sMsg = "L'impression a été annulée (par ex. par l'utilisateur) "
    sMsg = sMsg & "lors de l'impression ou du spoulage."
    Msgbox sMsg

  Case com.sun.star.view.PrintableState.JOB_FAILED
    sMsg = "L'impression a rencontré une erreur."
    Msgbox sMsg

  Case com.sun.star.view.PrintableState.JOB_SPOOLING_FAILED
    sMsg = "Le document a pu être imprimé, mais pas spoulé."
    Msgbox sMsg

 End Select

End sub
```

Référence de l'API en rapport :
http://api.openoffice.org/docs/common/ref/com/sun/star/document/OfficeDocument.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJobBroadcaster.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJobListener.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/PrintJobEvent.html
http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintJob.html

10.2.6 Exemple 3 : com.sun.star.awt.XKeyHandler

Les gestionnaires sont des types particuliers d'auditeurs. Comme les auditeurs, ils peuvent intercepter les événements mais en plus un gestionnaire agit comme un consommateur d'événements, en d'autres termes, un gestionnaire peut "avaler" des événements. A la différence des auditeurs, les méthodes d'un gestionnaire doivent avoir un résultat (booléen) : un résultat Vrai dit au broadcaster que l'événement est consommé par le gestionnaire, ceci fait que le broadcaster n'adressera pas l'événement aux gestionnaires subséquents.

Le gestionnaire com.sun.star.awt.XKeyHandler permet l'interception des événements clavier dans un document. L'exemple suivant montre un gestionnaire clavier qui agit en consommateur pour certains événements "touches appuyées" (touche “t”, “a”, “b”, “u”) :

```
Option Explicit
Global oDocView
Global oKeyHandler

Sub RegisterKeyHandler
 oDocView = ThisComponent.getCurrentController
 oKeyHandler = _
 createUnoListener("MyApp_", "com.sun.star.awt.XKeyHandler")

'   writedbginfo oKeyHandler

 oDocView.addKeyHandler(oKeyHandler)
End Sub

Sub UnregisterKeyHandler
 oDocView.removeKeyHandler(oKeyHandler)
End Sub

Sub MyApp_disposing(oEvt)
```

```
'on ne fait rien ici
End Sub

Function MyApp_KeyPressed(oEvt) As Boolean
  select case oEvt.KeyChar
    case "t", "a", "b", "u"
      MyApp_KeyPressed = True
      msgbox "La touche """ & oEvt.KeyChar & """ n'est pas autorisée !"
    case else
      MyApp_KeyPressed = False
  end select
End Function

Function MyApp_KeyReleased(oEvt) As Boolean
  MyApp_KeyReleased = False
End Function
```

Référence de l'API en rapport :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XUserInputInterception.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XExtendedToolkit.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XKeyHandler.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/KeyEvent.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/Key.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/KeyFunction.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/KeyModifier.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/InputEvent.html

10.2.7 Exemple 4 : com.sun.star.awt.XMouseClickHandler

Ce gestionnaire autorise la capture des clics de souris dans un document. Cet exemple montre une implémentation complète de ce gestionnaire :

```
Option Explicit

Global oDocView As Object
Global oMouseClickHandler As Object

Sub RegisterMouseClickHandler
  oDocView = ThisComponent.currentController
  oMouseClickHandler = _
  createUnoListener("MyApp_", "com.sun.star.awt.XMouseClickHandler")

'  writedbginfo oMouseClickHandler

  oDocView.addMouseClickHandler(oMouseClickHandler)

End Sub

Sub UnregisterMouseClickHandler
  on error resume next
  oDocView.removeMouseClickHandler(oMouseClickHandler)
  on error goto 0
End Sub



Sub MyApp_disposing(oEvt)
```

```
End Sub

Function MyApp_mousePressed(oEvt) As Boolean

 MyApp_mousePressed = False
End Function

Function MyApp_mouseReleased(oEvt) As Boolean
Dim sMsg As String

 With oEvt
  sMsg = sMsg & "Modifieurs = " & .Modifiers & Chr(10)
  sMsg = sMsg & "Boutons = " & .Buttons & Chr(10)
  sMsg = sMsg & "X = " & .X & Chr(10)
  sMsg = sMsg & "Y = " & .Y & Chr(10)
  sMsg = sMsg & "Nb de clics = " &  .ClickCount & Chr(10)
  sMsg = sMsg & "Déclenchement de Popup = " & .PopupTrigger '& Chr(10)
  'sMsg = sMsg & .Source.dbg_Methods
 End With

 ThisComponent.text.string = sMsg

 MyApp_mouseReleased = False
End Function
```

Référence de l'API en rapport :
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XUserInputInterception.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XMouseClickHandler.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/MouseEvent.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/MouseButton.html
http://api.openoffice.org/docs/common/ref/com/sun/star/awt/InputEvent.html

10.2.8 Exemple 5 : Liaison manuelle des évenements

Normalement, en programmation OOBasic, vous n'avez pas besoin d'auditeurs parce que vous pouvez lier manuellement un événement à une macro. Par exemple, depuis la boîte de dialogue “Adaptation” (menu “Outils”=>”Adaptation...”), sélectionner l'onglet "événements" vous permettra de lier des événements d'application ou de document. En outre, de nombreux objets pouvant être insérés dans un document ont une boîte de dialogue de propriétés avec un onglet "Evénements". Finalement, les boîtes de dialogue OOBasic et les contrôles font cela aussi bien.

Il est utile de noter que dans la liaison manuelle, le mécanisme sous-jacent est le même qu'avec les auditeurs, toutefois, vous pouvez ajouter des paramètres d'événement à vos macros pour obtenir des informations additionnelles sur l'événement.

Pour exécuter l'événement suivant, ouvrir un nouveau document Writer, ajouter un champ de texte et assigner manuellement la macro à l'événement "touche appuyée" du contrôle. Note : le nom de la macro et celui de l'événement sont arbitraires.

```
Option Explicit

' La macro est assignée manuellement à l'événement "Touche enfoncée"
' du contrôle champ de texte dans le document
Sub MyTextEdit_KeyPressed(oEvt)
Dim sMsg As String

 With oEvt
```

```
  sMsg = sMsg & "Modifieurs = " & .Modifiers & Chr(10)
  sMsg = sMsg & "Code touche = " & .KeyCode & Chr(10)
  sMsg = sMsg & "Caractère = " & .KeyChar & Chr(10)
  sMsg = sMsg & "Fonction = " & .KeyFunc & Chr(10)
  sMsg = sMsg & .Source.Dbg_supportedInterfaces
 End With

 msgbox sMsg
End Sub
```

11 Langage

11.1 Commentaires

C'est toujours une bonne méthode de commenter largement votre code. Ce qui est clair aujourd'hui ne le sera pas demain. L'apostrophe simple et le mot-clé REM indiquent tous deux le début d'un commentaire. Tout le texte suivant sera ignoré.

```
REM Ceci est un commentaire
REM Et voici un autre commentaire
' Et encore un autre commentaire
' Je pourrais continuer comme ça toute la journée
Dim i As Integer        REM i est utilisé comme variable index dans les boucles
Print i                 REM Ceci va imprimer la valeur de i
```

11.2 Variables

11.2.1 Noms

Les noms de variables sont limités à 255 caractères, ils doivent commencer par une lettre de l'alphabet non accentuée et peuvent contenir des chiffres. Les caractères souligné et espace sont aussi valides. Aucune distinction n'est faite entre les caractères majuscules et minuscules. Les noms de variables contenant des espaces doivent être placés entre crochets "[ ]".

11.2.2 Déclaration

Il est recommandé de déclarer vos variables avant de les utiliser. L'instruction "Option Explicit" vous oblige à le faire. Cette ligne doit se trouver dans votre code avant toute autre instruction. Si vous n'utilisez pas "Option Explicit", des variables mal écrites peuvent générer des bogues parfois difficiles à détecter (erreurs logicielles).

Utilisez Dim pour déclarer une variable. Voici la syntaxe de Dim :

[ReDim]Dim Nom1 [(*début* To *fin*)] [As *Type*][, Nom2 [(*début* To *fin*)] [As *Type*][,...]]

Cette syntaxe vous permet de déclarer plusieurs variables à la fois. *Nom* est un nom quelconque de variable ou de tableau. Les valeurs *début* et *fin* vont de -32768 à 32767. Elles définissent le nombre d'éléments (inclusivement), de sorte que *Nom1(début)* et *Nom1(fin)* sont tous deux des valeurs valides. Avec ReDim , les valeurs de *début* et *fin* peuvent être des expressions numériques. Les valeurs possibles pour *Type* sont Boolean, Currency, Date, Double, Integer, Long, Object, Single, String, et Variant.

Variant est le type par défaut si aucun type n'est spécifié, à moins que les commandes DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj, ou DefVar ne soient utilisées. Ces commandes vous permettent de définir le type de donnée d'après le premier caractère du nom de la variable.

Les variables String sont des chaînes de caractères, d'une longueur maximale de 64000 caractères.

Les variables Variant peuvent contenir tous les types et sont précisées à la définition.

Les variables Object doivent être suivies d'une instruction d'affectation Set.

L'exemple suivant illustre les problèmes qui peuvent arriver si vous ne déclarez pas vos variables. La variable non-définie "truc" sera par défaut de type Variant. Exécutez cette macro et voyez quels types Basic utilisera pour les variables non déclarées.

```
Sub TestNonDeclare
 print "1 :  ", TypeName(truc), truc
 truc= "ab217"
```

```
 print "2 :  ", TypeName(truc), truc
 truc= true
 print "3 :  ", TypeName(truc), truc
 truc= 5=5 ' devrait être un Booléen
 print "4 :  ", TypeName(truc), truc
 truc= 123.456
 print "5 :  ", TypeName(truc), truc
 truc=123
 print "6 :  ", TypeName(truc), truc
 truc= 1217568942 ' Pourrait être un Long
 print "7 :  ", TypeName(truc), truc
 truc= 123123123123123.1234 'Devrait être un Currency
 print "8 :  ", TypeName(truc), truc
End Sub
```

C'est un argument fort pour explicitement déclarer toutes les variables.

Attention On doit déclarer le type de variables individuellement pour chacune, sinon il sera par défaut de type Variant. “Dim a, b As Integer” déclare “a” comme type Variant et “b” comme type Integer.

```
Sub MultipleDeclaration
    Dim a, b As Integer
    Dim c As Long, d As String
    Dim e As Variant, f As Float
    Print TypeName(a)    Rem Empty , Variant par défaut
    Print TypeName(b)    Rem Integer, Déclaré Integer
    Print TypeName(c)    Rem Long, Déclaré Long
    Print TypeName(d)    Rem String, Déclaré String
    Print TypeName(e)    Rem Empty, Variant comme déclaré
    Print TypeName(f)    Rem Object, car Float est un type inconnu
    Print TypeName(g)    Rem Empty (Variant) car NON déclaré
End Sub
```

11.2.3 Les malfaisantes variables Global et les variables Static

Les variables globales sont habituellement déconseillées car elles peuvent être modifiées par n’importe quelle routine, n’importe où, et il est difficile de savoir quelle routine modifie quelle variable, si on les utilise. C’est pourquoi j'ai placé le terme “malfaisante” avant le terme “variable globale” lorsque j’enseignais à l’Université d’État de l’Ohio. C’était un moyen de rappeler à mes étudiants que, bien qu’il y ait des occasions pour utiliser des variables globales, ils devaient réfléchir avant de les utiliser.

Une variable globale doit être déclarée en dehors d’une procédure. On peut utiliser les mots-clés Public et Private pour préciser si la variable est globale à tous les modules ou seulement à ce module. En l’absence de Public ou Private, on suppose Private. La syntaxe est identique aux instructions Dim et ReDim.

Bien que les variables soient passées par référence si non stipulées autrement, les variables globales semblent être passées par valeur. Ce qui a causé au moins un bogue dans mon code.

A chaque appel d’une procédure, les variables locales à une procédure sont recréées. En déclarant la variable Static, elle gardera sa valeur. Dans l’exemple ci-dessous, le sous-programme Worker compte combien de fois il a été appelé. Rappel : les variables numériques sont initialisées à zéro et les chaînes de caractères sont initialisées à une chaîne vide.

```
Option Explicit
Public Auteur As String              REM Global à tous les modules
Private PrivateOne$        REM Global à ce module seulement
```

```
Dim PrivateTwo$        REM Global à ce module seulement


Sub PublicPrivateTest
  Author = "Andrew Pitonyak"
  PrivateOne$ = "Hello"
  Worker()
  Worker()
End Sub

Sub Worker()
      Static  Counter As Long          REM mémorise sa valeur entre deux appels
       Counter =  Counter + 1     REM compte chaque appel de Travailleur
      Print "Compteur = " +  Counter
      Print "Auteur = " + Author
End Sub
```

11.2.4 Types

D'un point de vue abstrait, le Basic OpenOffice.org supporte les types de variables numérique, chaîne, booléen, et objet. Les objets sont principalement utilisés pour se référer à des éléments internes comme des documents, des tables, etc... Avec un objet, vous pouvez utiliser les méthodes et les propriétés qui lui sont associées. Les types numériques sont initialisés à zéro et les chaînes de caractères sont initialisées à une chaîne vide "".

Pour connaître à l'exécution le type d'une variable, la fonction TypeName renvoie une chaîne de caractères désignant le type de la variable. La fonction VarType renvoie un entier Integer correspondant au type.

Mot-clé	***Type de variable***	***VarType***	***Auto Type***	***Defxxx***
Boolean	Booléen	11		DefBool
Currency	Monétaire avec 4 chiffres après la virgule	6	@	
Date	Date	7		DefDate
Double	Virgule flottante en double précision	5	#	DefDbl
Integer	Entier	2	%	DefInt
Long	Entier de grande valeur	3	&	DefLng
Object	Objet	9		DefObj
Single	Virgule flottante en simple précision	4	!	
String	Chaîne de caractères	8	$	
Variant	Peut contenir tous les types spécifiés par la définition	12		DefVar
(rien)	Variable non initialisée	0		
Null	Donnée invalide	1		

```
Sub ExampleTypes
      Dim b As Boolean
      Dim c As Currency
      Dim t As Date
      Dim d As Double
      Dim i As Integer
```

```
    Dim l As Long
    Dim o As Object
    Dim f As Single
    Dim s As String
    Dim v As Variant
    Dim n As Variant
    Dim x As Variant
    n = null
    x = f
    Print TypeName(b) + " " + VarType(b)   Rem Boolean 11
    Print TypeName(c) + " " + VarType(c)   Rem Currency 6
    Print TypeName(t) + " " + VarType(t)   Rem Date 7
    Print TypeName(d) + " " + VarType(d)   Rem Double 5
    Print TypeName(i) + " " + VarType(i)   Rem Integer 2
    Print TypeName(l) + " " + VarType(l)   Rem Long 3
    Print TypeName(o) + " " + VarType(o)   Rem Object 9
    Print TypeName(f) + " " + VarType(f)   Rem Single 4
    Print TypeName(s) + " " + VarType(s)   Rem String 8
    Print TypeName(v) + " " + VarType(v)   Rem Empty 0
    Print TypeName(n) + " " + VarType(n)   Rem Null 1
    Print TypeName(x) + " " + VarType(x)   Rem Single 4
End Sub
```

Variables booléennes Boolean

Bien que les variables Boolean utilisent les valeurs “True” ou “False” (pour Vrai et Faux respectivement), elles sont représentées en interne par les valeurs entières “-1” et “0” respectivement. Toute valeur numérique différente de 0 affectée à un booléen entraîne la valeur True. Voici des utilisations typiques :

```
Dim b as Boolean
b = True
b = False
b = (5 = 3)    'Affecte la valeur False
Print b        'Affiche 0
b = (5 < 7)    'Affecte la valeur True
Print b        'Affiche -1
b = 7          'Affecte la valeur True car 7 est différent de 0
```

Variables entières Integer

Les variables Integer sont des nombres entiers à 16 bits sur une plage de valeurs de -32768 à 32767. Si on affecte un nombre flottant à un Integer, la valeur est arrondie à l’entier le plus proche. Rajouter un caractère “%” derrière le nom d’une variable llui donne le type Integer.

```
Sub AssignFloatToInteger
  Dim i1 As Integer, i2%
  Dim f2 As Double
  f2= 3.5
  i1= f2
  Print i1  REM 4
  f2= 3.49
  i1= f2
  Print i1  REM 3
End Sub
```

Variables entières Long

Les variables Long sont des nombres entiers à 32 bits sur une plage de valeurs de -2.147.483.648 à 2.147.483.647. Si on affecte un nombre flottant à un Long, la valeur est arrondie à l'entier le plus proche. Rajouter un caractère "&" derrière le nom d'une variable lui donne le type Long.

```
Dim Age&
Dim Dogs As Long
```

Variables monétaires Currency

Les variables monétaires Currency sont des nombres à 64 bits en virgule fixe, avec 4 décimales et une partie entière de 15 chiffres. Ceci donne une plage de valeurs de -922.337.203.658.477,5808 à +922.337.203.658.477,5807. Rajouter un caractère "@" derrière le nom d'une variable lui donne le type Currency.

```
Dim Income@
Dim Cost As Currency
```

Variables flottantes Single

Les variables Single sont des nombres à 32 bits en virgule flottante. La plus grande valeur absolue est 3,402823 x 10E38. La plus petite valeur absolue non nulle est 1,401298 x 10E-45. Rajouter un caractère "!" derrière le nom d'une variable lui donne le type Single.

```
Dim Weight!
Dim Height As Single
```

Variables flottantes Double

Les variables Double sont des nombres à 64 bits en virgule flottante. La plus grande valeur absolue est 1,79769313486232 x 10E308. La plus petite valeur absolue non nulle est 4,94065645841247 x 10E-324. Rajouter un caractère "#" derrière le nom d'une variable lui donne le type Double.

```
Dim Weight#
Dim Height As Double
```

Variables de chaîne de caractères String

Les variables String nécessitent un caractère ASCII à un octet pour chaque caractère, et elles ont une longueur limitée à 64K octets. Rajouter un caractère "$" derrière le nom d'une variable lui donne le type String.

```
Dim FirstName$
Dim LastName As String
```

11.2.5 Object, Variant, Empty et Null

Les deux valeurs spéciales Empty et Null sont à considérer quand on manipule les types Object et Variant. La valeur Empty indique qu'aucune valeur n'a été assignée à la variable. Cela peut être testé avec la fonction IsEmpty(var). Le Null indique qu'aucune valeur valide n'est présente. Cela peut être testé avec la fonction IsNull(var).

Quand une variable de type Object est déclarée, elle contient la valeur Null. Quand une variable de type Variant est déclarée, elle contient la valeur Empty.

```
Sub ExampleObjVar
    Dim obj As Object, var As Variant
    Print IsNull(obj)        Rem True
    Print IsEmpty(obj)Rem False
    obj = CreateUnoService("com.sun.star.beans.Introspection")
    Print IsNull(obj)        Rem False
    obj = Null     '?? Vérifier sur la 1.0.3.1
```

```
    Print IsNull(obj)       Rem True

    Print IsNull(var)       Rem False
    Print IsEmpty(var)      Rem True
    var = obj
    Print IsNull(var)       Rem True
    Print IsEmpty(var)      Rem False
    var = 1
    Print IsNull(var)       Rem False
    Print IsEmpty(var)      Rem False
    var = Empty       Rem IsEmpty(Empty) échoue avec la 1.0.3.1 mais marche en 1.1 beta
    Print IsEmpty(var)      Rem True
End Sub
```

11.2.6 Dois-je utiliser Object ou Variant ?

En écrivant du code qui interagit avec les objets basic UNO, il faut décider quel type de variable utiliser. La plupart des exemples utilisent le type Object. Cependant, le guide du développeur, en page 132, suggère une autre déclaration.

Toujours utiliser le type variant pour les variables d'objets UNO et pas le type Object. Le type Object de OOBasic est conçu pour les objets OOBasic purs et pas pour UNO. Les variables Variant sont mieux pour les objets UNO, évitant des problèmes pouvant intervenir lors de comportements spécifiques au type object d'OOBasic.

```
Dim oService1 ' Ok
oService1 = CreateUnoService( "com.sun.star.anywhere.Something" )
Dim oService2 as Object ' NON recommandé
oService2 = CreateUnoService( "com.sun.star.anywhere.SomethingElse" )
```

Andreas Bregas ajoute que dans la plupart des cas, les 2 méthodes fonctionnent. Le guide du développeur préconise le type Variant car dans certains rares cas, l'utilisation du Type Objet conduit à une erreur à cause de la sémantique du type Object de l'ancien Basic. Mais si un programme en Basic utilise le type Objet et fonctionne correctement comme cela, il n'y a pas de problème.

11.2.7 Constantes

Le Basic OpenOffice.org reconnait les valeurs “True”(vrai), “False” (faux), et “PI”. Vous pouvez aussi définir vos propres constantes. On ne peut définir qu’une seule fois une constante. Les constantes n’ont pas de type, elles sont simplement insérées comme si on les avait tapées.

```
Const Gravity = 9.81
```

11.2.8 Tableaux

Un tableau vous permet de mettre de nombreuses valeurs différentes dans une seule variable. Par défaut, le premier élément est en position zéro. Mais vous pouvez préciser les positions de début et de fin. Voici quelques exemples :

```
Dim a(5) As Integer                 REM 6 éléments de 0 à 5 inclus
Dim b$(5 to 10) As String           REM 6 éléments de 5 à 10 inclus
Dim c(-5 to 5) As String            REM 11 éléments de -5 à 5 inclus
Dim d(5 To 10, 20 To 25) As Long    REM Tableau à deux dimensions
```

Si vous avez un tableau de Variant et que vous voulez le remplir rapidement, utilisez la fonction Array. Cela retournera un tableau de Variant avec des données incluses. C'est comme cela que j'établis une

liste de données.

```
Sub ExampleArray
  Dim a(), i%
  a = Array(0, 1, 2)
  a = Array("Zero", 1, 2.0, Now)
  REM String, Integer, Double, Date
  For i = LBound(a()) To UBound(a())
    Print TypeName(a(i))
  Next
End Sub
```

Option Base

Vous pouvez changer la valeur par défaut de la position basse, à 1 au lieu de zéro. Ceci doit être indiqué avant toute autre instruction exécutable du programme.

Syntaxe : Option Base { 0 | 1 }

LBound(NomDeTableau[,Dimension])

Renvoie la position basse du tableau. Le second paramètre optionnel est le numéro de la dimension du tableau pour laquelle vous désirez connaître la position basse ; ce numéro est compté à partir de 1 (et non pas zéro).

```
LBound(a())       REM 0
LBound(b$())      REM 5
LBound(c())       REM -5
LBound(d())       REM 5
LBound(d(), 1)    REM 5
LBound(d(), 2)    REM 20
```

UBound(NomDeTableau[,Dimension])

Renvoie la position haute du tableau. Le second paramètre optionnel est le numéro de la dimension du tableau pour laquelle vous désirez connaître la position haute ; ce numéro est compté à partir de 1 (et non pas zéro).

```
UBound(a())       REM 5
UBound(b$())      REM 10
UBound(c())       REM 5
UBound(d())       REM 10
UBound(d(), 1)    REM 10
UBound(d(), 2)    REM 25
```

Ce tableau est-il défini ?

Si un tableau est en réalité une liste vide, la position basse LBound aura une valeur supérieure à la position haute Ubound.

11.2.9 DimArray, changer la dimension

La fonction DimArray sert à définir ou à modifier le nombre de dimensions d'un tableau de Variant. DimArray(2, 2, 4) est équivalent à DIM a(2, 2, 4).

```
Sub ExampleDimArray
  Dim a(), i%
  a = Array(0, 1, 2)
  Print "" & LBound(a()) & " " & UBound(a())    REM 0 2
  a = DimArray()
  ' Empty array
  i = 4
  a = DimArray(3, i)
  Print "" & LBound(a(),1) & " " & UBound(a(),1)  REM 0, 3
  Print "" & LBound(a(),2) & " " & UBound(a(),2)  REM 0, 4
End Sub
```

11.2.10 ReDim, changer le nombre d'éléments

L'instruction ReDim est utilisée pour changer la taille d'un tableau.

```
Dim e() As Integer        REM taille non spécifiée
ReDim e(5) As Integer     REM positions 0 à 5 valides
ReDim e(10) As Integer    REM positions 0 à 10 valides
```

Le mot-clé Preserve peut être utilisé avec l'instruction ReDim pour préserver le contenu du tableau quand il est redimensionné.

```
Sub ReDimExample
    Dim a(5) As Integer
    Dim b()
    Dim c() As Integer
    a(0) = 0
    a(1) = 1
    a(2) = 2
    a(3) = 3
    a(4) = 4
    a(5) = 5
    Rem a est dimensionné de 0 à 5 avec a(i) = i
    PrintArray("a au début", a())
    Rem a est dimensionné de 1 à 3 avec a(i) = i
    ReDim Preserve a(1 To 3) As Integer
    PrintArray("a après ReDim", a())
    Rem Array() renvoie un type variant
    Rem b est dimensionné de 0 à 9 avec b(i) = i+1
    b = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    PrintArray("b a l'affectation initiale", b())
    Rem b est dimensionné de 1 à 3 avec b(i) = i+1
    ReDim Preserve b(1 To 3)
    PrintArray("b après ReDim", b())
    Rem Ce qui suit est NON valide car le tableau est déjà
    Rem dimensionné à une taille différente
    Rem a = Array(0, 1, 2, 3, 4, 5)

    Rem c est dimensionné de 0 à 5 avec a(i) = i
    Rem Si un "ReDim" avait été fait sur c, cela ne marcherait pas
    c = Array(0, 1, 2, 3, 4, 5)
    PrintArray("c, de type Integer après affectation", c())
    Rem Curieusement, ceci est autorisé mais c ne contiendra aucune donnée !
    ReDim Preserve c(1 To 3) As Integer
```

```
    PrintArray("c après ReDim", c())
End Sub

Sub PrintArray (lead$, a() As Variant)
    Dim i%, s$
    s$ = lead$ + Chr(13) + LBound(a()) + " to " + UBound(a()) + ":" + Chr(13)
    For i% = LBound(a()) To UBound(a())
        s$ = s$ + a(i%) + " "
    Next
    MsgBox s$
End Sub
```

La fonction Array mentionnée plus haut permet seulement de créer un tableau de Variant. Pour initialiser un tableau de type connu, vous pouvez utiliser la méthode suivante :

```
Sub ExampleSetIntArray
  Dim iA() As Integer
  SetIntArray(iA, Array(9, 8, 7, 6))
  PrintArray("", iA)
End Sub
Sub SetIntArray(iArray() As Integer, v() As Variant)
  Dim i As Long
  ReDim iArray(LBound(v()) To UBound(v())) As Integer
  For i = LBound(v) To UBound(v)
    iArray(i) = v(i)
  Next
End Sub
```

11.2.11 Tester les objets

Pour déterminer le type d'une variable, on peut utiliser les fonctions booléennes IsArray, IsDate, IsEmpty, IsMissing, IsNull, IsNumeric, IsObject et IsUnoStruct. La fonction IsArray renvoie True si le paramètre est un tableau. La fonction IsDate renvoie True s'il est possible de convertir l'objet en une Date. Une chaîne de caractères contenant une date correctement formatée renverra donc True pour la fonction IsDate. La fonction IsEmpty sert à tester si un objet de type Variant a été initialisé. La fonction IsMissing indique si un paramètre Optional est manquant. La fonction IsNull teste si un Variant contient la valeur spéciale Null, qui indique que la variable ne contient pas de donnée. La fonction IsNumeric sert à tester si une chaîne de caractères contient une valeur numérique. La fonction IsUnoStruct analyse la chaîne de caractères d'un nom de structure Uno et renvoie True si c'est un nom valide.

11.2.12 Opérateurs de comparaison

Les opérateurs de comparaison fonctionnent en général comme on s'y attend, mais ils n'effectuent pas une évaluation optimisée. On utilise les opérateurs de comparaison suivants :

Symbole	***Signification***
=	Egal à
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à

Symbole	*Signification*
<>	Différent de
Is	Est le même objet que

L'opérateur And réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. L'opérateur OR réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. L'opérateur XOR réalise une opération logique sur un type Boolean et une opération bit à bit sur les types numériques. Se rappeler qu'il s'agit du "OU exclusif". L'opérateur NOT réalise une opération logique de négation sur un type Boolean et une opération bit à bit sur les types numériques. Un simple test montre que les règles de priorité standard existent, à savoir que AND a une plus grande priorité que les opérateurs OR.

```
Option Explicit
Sub ConditionTest
    Dim msg As String
    msg = "AND possède une priorité "
    msg = msg & IIf(False OR True AND False, "égale", "supérieure")
    msg = msg & " à OR" & Chr(13) & "OR possède une priorité "
    msg = msg + IIF(True XOR True OR True, "supérieure", "inférieure ou égale ")
    msg = msg + "à XOR" + Chr(13)
    msg = msg & "XOR possède une priorité "
    msg = msg + IIF(True OR True XOR True, "supérieure", "inférieure ou égale ")
    msg = msg + "à OR"
    MsgBox msg
End Sub
```

11.3 Fonctions et Sous-programmes

Une fonction (Function) est un sous-programme (Sub) qui renvoie une valeur. Ce qui permet de l'utiliser dans une expression. Les sous-programmes et fonctions débutent ainsi :

Syntaxe de début : Function NomDeFonction[(Var1 [As Type][, Var2 [As Type][,...]]]) [As Type]
Syntaxe de début : Sub NomDeSousProg[(Var1 [As Type][, Var2 [As Type][,...]]])

Les fonctions déclarent un type de valeur de renvoi car elles renvoient une valeur. Pour affecter la valeur de renvoi, utilisez une instruction de la forme "NomDeFonction= valeur_renvoi". Vous pouvez réaliser plusieurs fois cette affectation mais seule la dernière sera renvoyée.

Utilisez une instruction Exit pour quitter immédiatement la procédure.

11.3.1 Paramètres optionnels

Un paramètre peut être déclaré optionnel avec le mot-clé Optional. La fonction IsMissing est alors utilisée pour savoir si un paramètre a été passé.

```
Sub testOptionalParameters()
  Print TestOpt()    REM MMM
  Print TestOpt(,)    REM MMM
  Print TestOpt(,,)    REM MMM
  Print TestOpt(1)    REM 1MM
  Print TestOpt(1,)  REM 1MM
  Print TestOpt(1,,)  REM 1MM
  Print TestOpt(1,2)  REM 12M
  Print TestOpt(1,2,)  REM 12M
```

```
 Print TestOpt(1,2,3)  REM 123
 Print TestOpt(1,,3)  REM 1M3
 Print TestOpt(,2,3)  REM M23
 Print TestOpt(,,3)  REM MM3
 Print TestOptI()    REM MMM
 Print TestOptI(,)    REM 488MM (Error)
 Print TestOptI(,,)  REM 488488M (Error)
 Print TestOptI(1)  REM 1MM
 Print TestOptI(1,)  REM 1MM
 Print TestOptI(1,,)  REM 1488M (Error)
 Print TestOptI(1,2)  REM 12M
 Print TestOptI(1,2,)  REM 12M
 Print TestOptI(1,2,3)REM 123
 Print TestOptI(1,,3)  REM 14883 (Error)
 Print TestOptI(,2,3)  REM 48823 (Error)
 Print TestOptI(,,3)  REM 4884883 (Error)
End Sub
Function TestOpt(Optional v1 As Variant, Optional v2 As Variant, Optional v3 As Variant) As String
 Dim s As String
 s = "" & IIF(IsMissing(v1), "M", Str(v1))
 s = s  & IIF(IsMissing(v2), "M", Str(v2))
 s = s  & IIF(IsMissing(v3), "M", Str(v3))
 TestOpt = s
End Function
Function TestOptI(Optional i1 As Integer, Optional i2 As Integer, Optional i3 As Integer) As String
 Dim s As String
 s = "" & IIF(IsMissing(i1), "M", Str(i1))
 s = s  & IIF(IsMissing(i2), "M", Str(i2))
 s = s  & IIF(IsMissing(i3), "M", Str(i3))
 TestOptI = s
End Function
```

Attention Sur la version 1.0.3.1, IsMissing ne fonctionne pas avec les paramètres Optional si le type n'est pas Variant et que le paramètre optionnel manquant est représenté par deux virgules consécutives. J'ai commencé à étudier ce comportement après discussion avec Christian Anderson [ca@ofs.no]. Voir le rapport 11678 dans Issuezilla.

11.3.2 Paramètres par référence ou par valeur

Avec une variable passée par valeur, je peux modifier la valeur du paramètre dans la procédure appelée et la variable originale ne changera pas. Par contre si la variable est passée par référence, alors en changeant la valeur du paramètre je change la variable originale. Le comportement par défaut est de passer par référence. Pour passer par valeur on doit utiliser le mot-clé ByVal avant la déclaration de paramètre. Si le paramètre est une constante, comme "4", et que vous le modifiez dans la procédure appelée, il peut ou non changer. Selon Andreas Bregas (ab@openoffice.org) c'est un bogue, que j'ai donc déclaré dans Issuezilla : (http://www.openoffice.org/project/www/issues/show_bug.cgi?id=12272).

```
Option Explicit
Sub LoopForever
    Dim I As Long
    I = 4
    LoopWorker(I)
    Print "I est passé par valeur et il est encore égal à " + I
    LoopForeverWorker(I)
```

```
    ' I vaut maintenant 1 donc on va afficher 1.
    Print "I est passé par référence et il vaut maintenant " + I
    ' Ceci va boucler à l'infini car 4 est une constante
    ' et on ne peut PAS la changer.
    Print "Passer par référence un paramètre constant, pour rire"
    Print LoopForeverWorker(4)
End Sub

Sub LoopWorker(ByVal n As Long)
    Do While n > 1
        Print n
        n = n - 1
    Loop
End Sub

Sub LoopForeverWorker(n As Long)
    Do While n > 1
        ' Ceci est amusant si n est une constante.
        Print n
        n = n - 1
    Loop
End Sub
```

11.3.3 Récursivité

Vos fonctions ne peuvent pas être récursives (c. à d. s'appeler elles-mêmes). Quand je dis “ne peuvent pas”, je devrais dire que vous ne devriez pas car vous n’obtiendrez pas les résultats que vous attendez. Il est prévu que cela change avec la version 1.1 (*NdT : confirmé*).

```
Option Explicit
Sub DoFact
  Print "Recursive = " + RecursiveFactorial(4)
  Print "Normal Factorial = " + Factorial(4)
End Sub

Function Factorial(n As Long) As Long
  Dim answer As Long
  Dim i As Long
  i = n
  answer = 1
  Do While i > 1
    answer = answer * i
    i = i - 1
  Loop
  Factorial = answer
End Function

' Ceci ne va pas marcher parce que vous ne pouvez pas utiliser la récursivité
Function RecursiveFactorial(n As Long) As Long
  If n > 2 Then
    RecursiveFactorial = n * RecursiveFactorial(n-1)
  Else
    RecursiveFactorial = 1
```

```
 End If
End Function
```

11.4 Contrôle du déroulement

11.4.1 If ...Then... Else...End If

On utilise la construction If pour exécuter un bloc de code en fonction de la valeur d'une expression booléenne. Bien qu'on puisse utiliser GoTo ou GoSub pour sortir d'un bloc If, on ne peut pas se brancher dans un bloc If.

Syntaxe :

If condition=vraie Then
bloc d'instructions
[ElseIf condition=vraie Then]
bloc d'instructions
[Else]
bloc d'instructions
End If

Syntaxe :

If condition=vraie Then instruction

Exemple :

```
If x < 0 Then
    MsgBox "Le nombre est négatif"
ElseIf x > 0 Then
    MsgBox " Le nombre est positif"
Else
    MsgBox "Le nombre est zéro"
End If
```

11.4.2 IIF

On utilise la construction IIF pour renvoyer une expression en fonction d'une condition. Ceci est similaire à la syntaxe du "?" en langage C.

Syntaxe : IIf (Condition, ExpressionSiVrai, ExpressionSiFaux)

Cette fonctionnalité est très similaire à ce code :

```
If (Condition) Then
    objet =  ExpressionSiVrai
Else
    objet =  ExpressionSiFaux
End If

max_age = IIf(Age_Jean > Age_Bernard, Age_Jean, Age_Bernard)
```

11.4.3 Choose

L'instruction Choose permet une sélection dans une liste selon un index.

Syntaxe : Choose (Index, Selection1[, Selection2, ... [,Selection_n]])

Si l'index vaut 1, l'élément Selection1 est renvoyé. Si l'index vaut 2, l'élément Selection2 est renvoyé. Vous pouvez deviner la suite !

11.4.4 For....Next

Cette structure répète un bloc d'instructions un nombre donné de fois.

Syntaxe :

For compteur = début To fin [Step incrément]
bloc d'instructions
[Exit For]
bloc d'instructions
Next [compteur]

La variable numérique "compteur" est initialisée à la valeur "départ". S'il n'y a pas de valeur "incrément", le compteur est incrémenté de 1 jusqu'à atteindre la valeur "fin". Si une valeur "incrément" est fournie, alors "incrément" est ajouté au compteur jusqu'à atteindre la valeur "fin". Le bloc d'instructions est exécuté une fois à chaque incrémentation.

Le "compteur" est optionnel sur l'instruction "Next" et réfère automatiquement à l'instruction "For" la plus récente.

On peut quitter prématurément une boucle "For" avec l'instruction "Exit For". Elle termine la boucle "For" la plus récente.

Exemple :

L'exemple suivant emploie deux boucles imbriquées pour trier un tableau d'entiers de 10 éléments (iValeurs()), qu'on a au préalable rempli avec un contenu varié :

```
Sub ForNextExampleSort
    Dim iEntry(10) As Integer
    Dim iCount As Integer, iCount2 As Integer, iTemp As Integer
    Dim bSomethingChanged As Boolean

    ' Remplir le tableau avec des entiers entre -10 et 10
    For iCount = LBound(iEntry()) To Ubound(iEntry())
        iEntry(iCount) = Int((20 * Rnd) -10)
    Next iCount

    ' Tri du tableau
    For iCount = LBound(iEntry()) To Ubound(iEntry())

        'Supposons le tableau trié
        bSomethingChanged  = False
        For iCount2 = iCount + 1 To Ubound(iEntry())
            If iEntry(iCount) > iEntry(iCount2) Then
                iTemp = iEntry(iCount)
                iEntry(iCount) = iEntry(iCount2)
                iEntry(iCount2) = iTemp
                bSomethingChanged  = True
            End If
        Next iCount2
        ' Si le tableau est déjà trié, arrêtons la boucle !
        If Not bSomethingChanged Then Exit For
    Next iCount
    For iCount = 1 To 10
        Print iEntry(iCount)
    Next iCount
```

```
End Sub
```

11.4.5 Boucle Do

L'aide en ligne contient une excellente et complète description , lisez là.

La structure de boucle Do a différentes formes et sert à répéter l'exécution d'un bloc de code tant qu'une condition est vraie. La forme la plus courante teste la condition avant le début de la boucle et répétera l'exécution du bloc d'instructions tant que la condition reste vraie. Si la condition est fausse au départ, la boucle ne sera jamais exécutée.

```
Do While condition
    Bloc d'instructions
Loop
```

Une forme similaire mais bien moins courante vérifie la condition avant le départ de la boucle et répétera l'exécution du bloc d'instructions tant que la condition reste fausse. Si la condition est vraie au départ, la boucle ne sera jamais exécutée.

```
Do Until condition
    Bloc d'instructions
Loop
```

On peut aussi placer le test en fin de boucle, dans ce cas le bloc d'instructions sera toujours exécuté au moins une fois. Pour exécuter toujours au moins une fois la boucle et répéter tant que la condition reste vraie, employer la structure suivante :

```
Do
    Bloc d'instructions
Loop While condition
```

Pour exécuter toujours au moins une fois la boucle et répéter tant que la condition reste fausse, employer la structure suivante :

```
Do
    Bloc d'instructions
Loop Until condition
```

On peut forcer la sortie d'une boucle Do avec l'instruction "Exit Do".

11.4.6 Select Case

L'instruction Select Case est similaire aux instructions "case" et "switch" dans d'autres langages. Elle émule de multiples blocs "Else If" dans une instruction "If". On spécifie une unique expression conditionnelle, qui est comparée à plusieurs valeurs pour chercher une égalité comme suit :

```
Select Case  variable
    Case expression1
        bloc d'instructions 1
    Case expression2
        bloc d'instructions 2
    Case Else
        bloc d'instructions 3
End Select
```

La valeur de la variable est comparée dans chaque instruction Case. Je ne connais pas de limitation de type autre que la compatibilité entre le type de la variable et le type de l'expression. Le premier bloc d'instruction correspondant est exécuté. Si aucune condition ne correspond, le bloc optionnel Case Else est exécuté.

Expressions Case

Une expression Case est habituellement une constante comme "Case 4" ou "Case Hello". On peut indiquer des valeurs multiples en les séparant par des virgules : "Case 3, 5, 7". Pour tester une plage de

valeurs, il existe le mot-clé “To” , exemple “Case 5 To 10”. On peut tester des séries ouvertes de valeurs avec “Case < 10” ou avec le mot-clé “Is”, exemple “Case Is < 10”.

Attention Faites attention quand vous utilisez une plage de valeurs dans une instruction Case. L'aide en ligne contient souvent des exemples incorrects, comme "Case i > 2 AND i < 10". C'est difficile à comprendre et à programmer correctement.

Exemple incorrect simple

J'ai vu beaucoup d'exemples incorrects, aussi je vais prendre le temps de vous montrer quelques choses qui ne fonctionnent pas. Je commencerai par un exemple très simple :

Correct	***Correct***	***Incorrect***
i = 2 Select Case i Case 2	i = 2 Select Case i Case Is = 2	i = 2 Select Case i Case i = 2

L'exemple incorrect échoue car " Case i = 2" est la réduction de "Case Is = (i = 2)". L'expression (i=2) est évaluée comme True, soit -1. Cela revient à évaluer l'expression " Case Is = -1" dans cet exemple.

Si vous comprenez cet exemple incorrect simple, alors vous êtes prêt des exemples difficiles.

Exemple incorrect avec une plage

L'exemple suivant est dans l'aide en ligne.

```
Case Is > 8 AND iVar < 11
```

Cela ne fonctionne pas car c'est évalué comme :

```
Case Is > (8 AND (iVar < 11))
```

L'expression (iVar<11) est évaluée comme True ou False. Souvenez vous que True=-1 et False=0. L'opérateur AND est appliqué, bit à bit, entre 8 et -1 (True) ou 0 (False), avec comme résultat soit 8 soit 0. L'expression est donc réduite à une des 2 expressions.

Si iVar est plus petite que 11 :

```
Case Is > 8
```

Si iVar est supérieure ou égale à 11 :

```
Case Is > 0
```

Exemple incorrect avec une plage

J'ai aussi vu cet exemple incorrect.

```
Case  i > 2 AND i < 10
```

Ceci ne fonctionne pas car c'est évalué comme :

```
Case  Is = (i > 2 AND i < 10)
```

Les plages, La Voie Correcte

L'expression

```
Case   Expression
```

est probablement correcte si elle peut être écrite

```
Case Is = (Expression)
```

Ma solution initiale était :

```
Case  iif(Boolean Expression, i, i+1)
```

J'étais fier de moi jusqu'à ce que Bernard Marcelly me donne la brillante solution suivante :

```
Case  i XOR NOT (Boolean Expression)
```

Après ma confusion initiale, j'ai réalisé combien c'était réellement brillant. Ne pas tenter de simplifier à la réduction évidente "i AND ()" car cela échoue si i = 0. J'ai fait cette erreur.

```
Sub DemoSelectCase
    Dim i%

    i = Int((15 * Rnd) -2)
    Select Case i%
        Case 1 To 5
            Print "Nombre entre 1 et 5"
        Case 6, 7, 8
            Print " Nombre entre 6 et 8"

        Case IIf(i > 8 AND i < 11, i, i+1)
            Print "Supérieur à 8"
        Case i% XOR NOT(i% > 8 AND i% < 11 )
            Print i%, "Nombre entre 9 et 10"
        Case Else
            Print "En dehors de la plage 1 à 10"
        End Select
End Sub
```

11.4.7 While...Wend

Il n'y a rien de spécial dans la structure While...Wend, qui a la forme suivante :

```
While Condition
    Bloc d'instructions
Wend
```

Conseil Cette structure a des limitations qui n'existent pas dans la structure Do While...Loop et n'apporte aucun avantage particulier. On ne peut pas utiliser l'instruction Exit, ni sortir par un GoTo.

11.4.8 GoSub

L'instruction GoSub provoque un saut à une étiquette de sous-programme définie dans le sous-programme en cours. Quand l'instruction Return est atteinte, l'exécution continue à partir du point d'appel initial. Si une instruction Return est rencontrée sans qu'un Gosub n'ait été préalablement effectué, une erreur est générée. Autrement dit, Return n'est pas un équivalent de Exit Sub ni de l'instruction Exit. On estime généralement que l'utilisation de fonctions et de sous-programmes produit un code plus compréhensible que l'utilisation de GoSub et GoTo.

```
Option Explicit
Sub ExampleGoSub
    Dim i As Integer
    GoSub Line2
    GoSub Line1
    MsgBox "i = " + i, 0, "Exemple de GoSub"
    Exit Sub
Line1:
    i = i + 1
    Return
Line2:
    i = 1
    Return
```

```
End Sub
```

Conseil GoSub est un vestige persistant des vieux dialectes BASIC, gardé par souci de compatibilité. GoSub est fortement déconseillé parce qu'il tend à produire un code illisible. L'utilisation de sous-programmes et fonctions est préférable.

11.4.9 GoTo

L'instruction GoTo provoque un saut à une étiquette définie dans le sous-programme en cours. On ne peut pas sauter à l'extérieur du sous-programme en cours.

```
Sub ExampleGoTo
    Dim i As Integer
    GoTo Line2
Line1:
    i = i + 1
    GoTo TheEnd
Line2:
    i = 1
    GoTo Line1
TheEnd:
    MsgBox "i = " + i, 0, "Exemple de GoTo"
End Sub
```

Conseil GoTo est un vestige persistant des vieux dialectes BASIC, gardé par souci de compatibilité. GoTo est fortement déconseillé parce qu'il tend à produire un code illisible. L'utilisation de sous-programmes et fonctions est préférable.

11.4.10 On GoTo

Syntaxe : On N GoSub Etiquette1[, Etiquette2[, Etiquette3[,...]]]
Syntaxe : On N GoTo Etiquette1[, Etiquette2[, Etiquette3[,...]]]

Ceci fait sauter l'exécution à une étiquette selon la valeur de l'expression numérique *N*. Il n'y a pas de saut si N vaut zéro. L'expression numérique *N* doit être dans la plage de valeurs 0 à 255. Ceci est similaire aux instructions "computed goto", "case" et "switch" d'autres langages. Ne pas essayer de sauter à l'extérieur du sous-programme ou de la fonction en cours.

```
Option Explicit
Sub ExampleOnGoTo
    Dim i As Integer
    Dim s As String
    i = 1
    On i+1 GoSub Sub1, Sub2
    s = s & Chr(13)
    On i GoTo Line1, Line2
    REM Cet Exit provoque la sortie si on ne continue pas l'exécution
    Exit Sub
Sub1:
    s = s & "Dans Sub 1" : Return
Sub2:
    s = s & "Dans Sub 2" : Return
Line1:
    s = s & "A  Ligne1" : GoTo TheEnd
Line2:
    s = s & "A Ligne2"
TheEnd:
```

```
    MsgBox s, 0, "Exemple de On GoTo"
End Sub
```

11.4.11 Exit

L'instruction Exit permet de sortir d'une boucle Do Loop ou For Next, d'une Function ou d'un Sub. L'utilisation de l'instruction Exit doit apparaître dans les structures du code qu'elle est censée contrôler sous peine de générer une erreur (par exemple, l'instruction Exit For ne peut être utilisée qu'à l'intérieur d'une boucle For). Les différentes formes sont les suivantes :

- Exit DO Continue l'exécution après la prochaine instruction Loop.
- Exit For Continue l'exécution après la prochaine instruction Next.
- Exit Function Sort immédiatement de la fonction en cours.
- Exit Sub Sort immédiatement de la Sub en cours .

```
Option Explicit
Sub  ExitExample
    Dim a%(100)
    Dim i%
    REM Remplir le tableau avec 100, 99, 98, ..., 0
    For i = LBound(a()) To UBound(a())
        a(i) = 100 - i
    Next i
    Print SearchIntegerArray(a(), 0  )
    Print SearchIntegerArray(a(), 10 )
    Print SearchIntegerArray(a(), 100)
    Print SearchIntegerArray(a(), 200)
End Sub

Function SearchIntegerArray( list(), num%) As Integer
    Dim i As Integer
    SearchIntegerArray = -1
    For i = LBound(list) To UBound(list)
        If list(i) = num Then
            SearchIntegerArray = i
            Exit For
        End If
    Next i
End Function
```

11.4.12 Gestion d'erreurs

Vos macros peuvent rencontrer plusieurs types d'erreurs. Certaines erreurs sont à gérer (comme un fichier manquant) et d'autres simplement à ignorer. Les erreurs dans les macros sont traitées par l'instruction :

On [Local] {Error GoTo Labelname | GoTo 0 | Resume Next}

On Error permet de spécifier comment les erreurs doivent être gérées, avec la possibilité de définir votre propre gestionnaire d'erreur. Si "Local" est utilisé, la gestion d'erreur n'est active que pour la procédure ou fonction courante, sinon elle s'applique au module entier.

Conseil	Une procédure peut contenir plusieurs gestions d'erreurs. Chaque On Error peut traiter les erreurs différemment (l'aide en ligne est fausse quand elle dit qu'une gestion d'erreurs doit apparaître en début de procédure).

Spécifier comment gérer une erreur

Pour ignorer toutes les erreurs, utiliser “On Error Resume Next”. Quand une erreur apparaît, cette commande impliquera qu'elle sera ignorée et l'instruction suivante sera exécutée.

Pour spécifier votre propre gestionnaire d'erreur, utiliser “On Error GoTo Label”. Pour définir un Label dans OOBasic, taper du texte sur une ligne seule suivi de deux-points. Les labels doivent être uniques. Quand une erreur sera générée, l'exécution de la macro sera transférée à la position du label.

Vous pouvez annuler une gestion d'erreur en utilisant “On Error GoTo 0”. Quand une erreur apparaîtra votre gestionnaire d'erreur ne sera plus appelé. Ceci est différent de “On Error Resume Next”. Cela implique qu'à la prochaine erreur rencontrée, OOBasic stoppera son exécution comme cela est fait par défaut(arrêt de l'exécution de la macro avec un message d'erreur).

Écrire le gestionnaire d'erreur

Quand une erreur apparaît et que l'exécution est transférée à votre gestionnaire, voici quelques fonctions utiles pour déterminer ce qui s'est passé et où.

- Error([num]) : Renvoie le message d'erreur en tant que chaîne de caractères. Vous pouvez, en option, indiquer un numéro d'erreur spécifique pour récupérer sa signification. Ces textes sont localisés.
- Err() : Retourne le numéro de la dernière erreur.
- Erl() : Indique le numéro de ligne de la dernière erreur.

Une fois l'erreur gérée, il faut décider comment continuer.

- Rien et laisser l'exécution se poursuivre.
- Sortir de la fontion ou du sous-programme en utilisant “Exit Sub” ou “Exit Function”.
- Utiliser “Resume” pour exécuter à nouveau la même ligne. Prudence avec ça ! Si la gestion d'erreur n'a pas corrigé l'erreur vous entrerez dans une boucle infinie.

```
Sub ExampleResume
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
    Print x
    Exit Sub
oopsy:
    y = 2
    Resume
End Sub
```

- Utiliser “Resume Next” pour poursuivre l'exécution à la ligne suivant celle qui a généré l'erreur.

```
Sub ExampleResumeNext
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
    Print x
    Exit Sub
oopsy:
    x = 7
    Resume Next
```

```
End Sub
```

- Utiliser “Resume Label:” pour poursuivre l'exécution à un label spécifique.

```
Sub ExampleResumeLabel
    Dim x%, y%
    x = 4 : y = 0
    On Local Error Goto oopsy
    x = x / y
GoHere:
    Print x
    Exit Sub
oopsy:
    x = 7
    Resume GoHere:
End Sub
```

Un exemple

La macro suivante illustre quelques techniques excellentes de gestion des erreurs :

```
'*********************************************************************
'Auteur : Bernard Marcelly
'email :   marcelly@club-internet.fr
Sub ErrorHandlingExample
    Dim v1 As Double
    Dim v2 As Double
    Dim v0 As Double

    On Error GoTo TreatError1
    v0 = 0 : v1= 45 : v2= -123 ' initialisations
    v2= v1 / v0 ' division par 0 => erreur
    Print "Result1:", v2

    On Error Goto TreatError2 ' change le gestionnaire d'erreur
    v2= 456 ' initialisation
    v2= v1 / v0 ' 'division par 0 => erreur
    Print "Result2:", v2 ' ne sera pas executé

Label2:
    Print "Result3:", v2 ' atteint par la gestion d'erreur

    On Error Resume Next ' ignore toute erreur
    v2= 963 ' initialisation
    v2= v1 / v0 '  division par 0 => erreur
    Print "Result4:", v2 ' sera executé

    On Error Goto 0 ' désactive la gestion d'erreur en cours
    rem La gestion standard est désormais active
    v2= 147 ' initialisation
    v2= v1 / v0 ' division par 0 => erreur
    Print "Result5:", v2 ' ne sera pas executé
    Exit Sub

TreatError1:
    Print "TreatError1 : ", error
```

```
    v2= 0
    Resume Next ' continue après l'instruction en erreur

TreatError2:
    Print "TreatError2 : line ", erl, "error number", err
    v2= 123456789
    Resume Label2
End Sub
```

11.5 Divers

Ce chapitre contient diverses choses que je connais seulement à travers des exemples, mais dont je n'ai pas trouvé l'utilité.

****

On peut mettre plusieurs instructions sur la même ligne en les séparant par un " : " (deux-points).

****

Avec une instruction sur une ligne, la structure If Then n'a pas besoin d'être fermée par un End If.

```
Sub SimpleIf
    If 4 = 4 Then Print "4 = 4" : Print "Hello"  Rem Va s'afficher
    If 3 = 2 Then Print "3 = 2"                  Rem Ne va pas s'afficher parce que 3 <> 2
End Sub
```

****

Librairies, dialogues, IDE, Import et Export de Macros.

With object ... End With

****

Comment démarrer à partir de la ligne de commande ?

Le paramètre de lancement de macro s'écrit :

soffice.exe macro:/library module macro

Exemple :

soffice.exe macro:///standard.module1.macro1

Mais attention ! Si la macro ne fait rien ou n'ouvre rien dans le document, la macro est exécutée puis OpenOffice.org est fermé.

****

La copie d'un Object copie seulement la référence. La copie d'une structure réalise une nouvelle copie. Voir EqualUnoObjects pour un exemple.?? Ceci peut poser un problème et alors l'objet devra être recopié!

12 Opérateurs et priorités

OpenOffice.org Basic supporte les opérateurs numérique de base -, +, /, *, et ^. Les opérateurs suivent l'ordre de priorité standard mais je l'indique ici aussi. Les opérateurs logiques renvoient 0 pour faux (pas de bits positionnés) et -1 pour vrai (tous les bits sont positionnés). Pour une information plus complète, voyez la section listant opérateurs et fonctions.

Priorité	*Opérateur*	*Description*
0	AND	Bit à bit pour les numériques et logique pour les Booléens
0	OR	Bit à bit pour les numériques et logique pour les Booléens
0	XOR	Bit à bit pour les numériques et logique pour les Booléens
0	EQV	Équivalence logique et/ou au niveau du bit
0	IMP	Implication logique (bogué depuis 1.0.3.1)
1	=	Logique
1	<	Logique
1	>	Logique
1	<=	Logique
1	>=	Logique
1	<>	Logique
2	-	Soustraction numérique
2	+	Addition numérique et concaténation de chaînes
2	&	Concaténation de chaînes
3	*	Multiplication numérique
3	/	Division numérique
3	MOD	Reste numérique après division entière
4	^	Puissance numérique

```
Sub TestPrecedence
    Dim i%
    Print 1 + 2 OR 1        REM  imprime  3
    Print 1 + (2 OR 1)      REM  imprime  4
    Print 1 + 2 AND 1       REM  imprime  1
    Print 1 + 2 * 3         REM  imprime  7
    Print 1 + 2 * 3 ^2      REM  imprime 19
    Print 1 = 2 OR 4        REM  imprime  4
    Print 4 AND 1 = 1       REM  imprime  4
End Sub
```

Attention	Les valeurs booléennes sont stockées en interne comme des entiers valant 0 pour False et -1 pour True. Ceci permet l'emploi des opérateurs numériques avec des valeurs booléennes mais je le déconseille (1 + Vrai = Faux). Utilisez plutôt les opérateurs booléens.

13 Manipulations de chaînes de caractères

OOBasic fournit quelques fonctions de manipulation des chaînes de caractères.

Fonction	*Description*
Asc(s$)	Valeur ASCII du premier caractère
Chr$(i)	Caractère correspondant à une valeur ASCII
CStr(Expression)	Convertit une expression numérique en chaîne de caractères
Format(number [, f])	Formate un nombre sur la base d'une chaîne de formatage
Hex(Number)	Chaîne représentant la valeur hexadécimale d'un nombre
InStr([i,] s$, f$[, c])	Position de f dans s, 0 si non trouvé. Peut être insensible à la casse (c=1). Le type retourné est un Long mis dans un Integer donc la valeur retournée peut être négative pour les grandes chaînes.
LCase(s$)	Met en minuscule
Left(s$, n)	Renvoie les n premiers caractères en partant de la gauche. n est un Integer alors que la chaîne peut être d'une taille de 64K
Len(s$)	Renvoie le nombre de caractères de la chaîne
LSet s$ = Text	Aligne une chaîne à gauche. ??Bogue dans la 1.0.3.1, supposé fixé dans la 1.1
LTrim(s$)	Renvoie une chaîne sans espace à gauche, ne modifie pas la chaîne
Mid(s$, i[, n])	Sous-chaîne à partir de la position i sur une longueur de n
Mid(s$, i, n, r$)	Remplace la sous-chaîne par r avec des limitations. Je l'emploie pour supprimer des parties.
Oct(Number)	Chaîne représentant la valeur Octale d'un nombre
Right(s$, n)	Renvoie les n derniers caractères de la chaîne. n est un Integer alors que la chaîne peut être d'une taille de 64K
RSet s$ = Text	Aligne une chaîne à droite
RTrim(s$)	Renvoie une chaîne sans espaces à la fin
Space(n)	Retourne une chaîne avec le nombre d'espaces spécifié
Str(Expression)	Convertit l'expression numérique en chaîne
StrComp(x$, y$[, c])	Retourne -1 si x>y, 0 si x=y et 1 if x<y. Si c=1 alors insensible à la casse
String(n, {i | s$})	Crée une chaîne de n caractères. Si un entier est donné, il est considéré comme la valeur ASCII du caractère à répéter. Si une chaîne est spécifiée, son premier caractère est répété n fois.
Trim(s$)	Renvoie une chaîne débarrassée de ses espaces de début et de fin
UCase(s$)	Renvoie la chaîne en majuscule
Val(s$)	Convertit la chaîne en nombre

Dans l'aide en ligne, l'exemple de changement de la casse d'une chaîne de caractères est erroné. Voici comment il faut lire :

```
Sub ExampleLUCase
    Dim sVar As String
    sVar = "Las Vegas"
    Print LCase(sVar) REM Affiche "las vegas"
```

```
    Print UCase(sVar) REM Affiche "LAS VEGAS"
end Sub
```

13.1 Enlever des caractères d'une Chaîne

Cette macro retire des caractères d'une chaîne. Cette fonctionnalité aurait pu être assurée par la fonction Mid citée précédemment :

```
'Retire un certain nombre de caractères d'une chaîne
Function RemoveFromString(s$, index&, num&) As String
    If num = 0 Or Len(s) < index Then
        'Si on ne retire rien ou en dehors de la taille de la chaîne, on retourne la chaîne initiale
        RemoveFromString = s
    ElseIf index <= 1 Then
        'Retire à partir de début
        If num >= Len(s) Then
            RemoveFromString = ""
        Else
            RemoveFromString = Right(s, Len(s) - num)
        End If
    Else
        'Retire du milieu
        If index + num > Len(s) Then
            RemoveFromString = Left(s,index - 1)
        Else
            RemoveFromString = Left(s,index - 1) + Right(s, Len(s) - index - num + 1)
        End If
    End If
End Function
```

13.2 Remplacer du texte dans une chaîne de caractères

Cette macro pourrait être utilisée pour effacer des zones d'une chaîne en spécifiant la chaîne de remplacement comme une chaîne vide. Ma première idée a été d'utiliser la fonction Mid() pour ça aussi, mais il est apparu que la fonction Mid() ne permet pas de rendre la chaîne initiale plus grande que ce qu'elle est. A cause de ça, j'ai dû écrire cette macro. Elle ne modifie pas la chaîne source mais en crée une nouvelle avec l'occurrence remplacée.

```
Rem s$ chaine source à modifier
Rem index entier long indiquant où le remplacement doit avoir lieu (Base 1)
Rem         Si index  <= 1 le texte est inséré au debut de la chaîne
Rem         Si index > Len(s) le texte est inséré en fin de chaîne
Rem num est un entier long indiquant combien de caractères sont à remplacer
Rem         Si num=0, rien n'est retiré mais la nouvelle chaîne est insérée
Rem (replaces) est la chaine de remplacement.
Function ReplaceInString(s$, index&, num&, replaces$) As String
    If index <= 1 Then
        'Place en début de chaine
        If num < 1 Then
            ReplaceInString = replaces + s
        ElseIf num > Len(s) Then
            ReplaceInString = replaces
        Else
            ReplaceInString = replaces + Right(s, Len(s) - num)
        End If
    ElseIf index + num > Len(s) Then
```

```
        ReplaceInString = Left(s,index - 1) + replaces
    Else
        ReplaceInString  = Left(s,index - 1) + replaces + Right(s, Len(s) - index - num + 1)
    End If
End Function
```

13.3 Afficher les valeurs ASCII d'une Chaîne de caractères

Cette macro pourrait paraître bizarre mais je l'ai utilisée pour décider comment un texte était stocké dans un document. Elle affiche la chaîne de caractères complète sous forme d'une suite de codes ASCII :

```
Sub PrintAll
    PrintAscii(ThisComponent.text.getString())
End Sub
Sub PrintAscii(TheText As String)
    If Len(TheText) < 1 Then Exit Sub
    Dim msg$, i%
    msg = ""
    For i = 1 To Len(TheText)
        msg = msg + Asc(Mid(TheText,i,1)) + " "
    Next i
    Print msg
End Sub
```

13.4 Supprimer toutes les occurrences d'une chaîne de caractères

```
Rem efface toutes les occurences bad$ de s$
Rem modifie la chaîne s$
Sub RemoveFromString(s$, bad$)
    Dim i%
    i = InStr(s, bad)
    Do While i > 0
        Mid(s, i, Len(bad), "")
        i = InStr(i, s, bad)
    Loop
End Sub
```

14 Manipulations numériques

Fonction	*Description*
Abs(Number)	Valeur absolue de type double
Asc(s$)	Valeur ASCII du premier caractère
Atn(x)	Retourne l'angle en radians dont la tangente est x
Blue(color)	Composante bleue d'un code de couleur
CByte(Expression)	Convertit en Byte une chaîne de caractères ou un nombre
CDbl(Expression)	Convertit en Double une chaîne de caractères ou un nombre
CInt(Expression)	Convertit en Integer une chaîne de caractères ou un nombre
CLng(Expression)	Convertit en Long une chaîne de caractères ou un nombre
Cos(x)	Cosinus de l'angle spécifié en radians
CSng(Expression)	Convertit en Single précision une chaîne de caractères ou un nombre
CStr(Expression)	Convertit en String une chaîne de caractères ou un nombre
Exp(Expression)	Fonction exponentielle, base e = 2.718282, inverse de Log()
Fix(Expression)	Partie entière après troncature
Format(number [, f$])	Formate un nombre suivant une chaîne de formatage
Green(color)	Composante verte d'un code de couleur
Hex(Number)	Chaîne représentant la valeur Hexadécimale d'un nombre
Int(Number)	Arrondit à la partie entière. Voir également : Fix()
IsNumeric (Var)	Teste si l'expression est un nombre
Log(Number)	Logarithme naturel (neperien) d'un nombre
Oct(Number)	Chaîne représentant la valeur en Octale d'un nombre
Randomize [Number]	Initialise le générateur de nombre aléatoire
Red(color)	Composante rouge d'un code couleur
RGB (Red, Green, Blue)	Valeur de couleur (Long) composée des valeurs rouges, vertes, bleues
Rnd [(Expression)]	Nombre aléatoire entre 0 et 1
Sgn (Number)	Retourne 1, -1, or 0 si le nombre est positif, négatif ou nul
Sin(x)	Sinus de l'angle spécifié en radians
Sqr(Number)	Racine carrée d'un nombre
Tan(x)	Tangente de l'angle spécifié en radians
n = TwipsPerPixelX	Nombre de twips pour la largeur d'un pixel
n = TwipsPerPixelY	Nombre de twips pour la hauteur d'un pixel
Val(s$)	Convertit une chaîne en nombre

15 Manipulations de dates

Fonction	*Description*
CDate(Expression)	Convertit en date un nombre ou une chaîne
CDateFromIso(String)	Retourne la date numérique à partir d'une chaîne contenant une date au format ISO
CDateToIso(Number)	Retourne date ISO à partir d'une date générée par DateSerial ou DateValue
Date	Date système en cours
Date = s$	Change la date système
DateSerial(y%, m%, d%)	Date à partir des arguments Année, Mois, Jour
DateValue([date])	Entier long utilisable pour calculer des différences de date
Day(Number)	Jour du mois d'une date obtenue avec DateSerial ou DateValue
GetSystemTicks()	Retourne les tics systèmes fourni par le système d'exploitation
Hour(Number)	Heures d'un temps obtenu par TimeSerial ou TimeValue
Minute(Number)	Minutes d'un temps obtenu par TimeSerial ou TimeValue
Month(Number)	Mois d'une date obtenue par DateSerial ou DateValue
Now	Date et heure système courantes
Second(Number)	Secondes d'une date obtenue par TimeSerial or TimeValue
Time	Heure système en cours
Timer	Nombre de secondes écoulées depuis minuit
TimeSerial (h, m, s)	Heure numérique à partir des paramètres Heure, Minutes, Secondes
TimeValue (s$)	Heure numérique à partir d'une chaîne formatée
Wait millisec	Pause pendant le nombre de millisecondes spécifiées
WeekDay(Number)	Jour de la semaine d'une date obtenue par DateSerial ou DateValue
Year(Number)	Année d'une date obtenue par DateSerial or DateValue

16 Manipulations de fichiers

Fonction	*Description*
ChDir (s$)	Change le répertoire ou le lecteur courant
ChDrive(s$)	Change le lecteur courant
Close #n% [, #n2%[,...]]	Ferme les fichiers ouverts avec l'instruction Open
ConvertFromURL(s$)	Convertit une URL en nom de fichier système
ConvertToURL(s$)	Convertit un nom de fichier système en URL
CurDir([s$])	Répertoire courant du lecteur spécifié
Dir [s$ [, Attrib%]]	Listing du répertoire
EOF(n%)	Le pointeur de fichier a-t-il atteint la fin du fichier ?
FileAttr (n%, Attribut%)	Attributs d'un fichier ouvert
FileCopy from$, to$	Copie un fichier
FileDateTime(s$)	Chaîne Date et Heure du fichier
FileExists(s$)	Un fichier ou répertoire existe-t-il ?
FileLen (s$)	Longueur du fichier en octets
FreeFile	Numéro suivant disponible de fichier. Évite les utilisations simultanées.
Get [#]n%, [Pos], v	Lit un enregistrement ou un nombre d'octets d'un fichier
GetAttr(s$)	Retourne une séquence identifiant le type de fichier
Input #n% v1[, v2[, [,...]]	Lit des données d'un fichier ouvert en séquentiel
Kill f$	Efface un fichier du disque
Line Input #n%, v$	Affecte une ligne d'un fichier séquentiel dans une variable
Loc (FileNumber)	Position courante dans un fichier
Lof (FileNumber)	Taille courante du fichier
MkDir s$	Crée un nouveau répertoire
Name old$, new$	Renomme un fichier ou un répertoire existant
Open s$ [#]n%	Ouverture d'un fichier. La plupart des paramètres ne sont pas listés, c'est très souple.
Put [#] n%, [pos], v	Écrit un enregistrement ou une séquence d'octets dans un fichier
Reset	Ferme tous les fichiers et vide tous les buffers sur le disque
RmDir f$	Supprime un répertoire
Seek[#]n%, Pos	Bouge le pointeur de fichier
SetAttr f$, Attribute%	Définit les attributs du fichier
Write [#]n%, [Exprs]	Écrit des données dans un fichier séquentiel

17Opérateurs, instructions et fonctions

Le but de ce chapitre n'est pas de reprendre mot pour mot l'aide en ligne mais d'apporter quelques remarques complémentaires concernant les mots réservés du langage.

17.1Description :

Retranche deux valeurs numériques. La préséance mathématique usuelle des opérateurs est applicable comme décrite page164..

Syntaxe :

Result = Expression1 - Expression2

Paramètres :

Result : Résultat de la soustraction.

Expression1, Expression2 : Toute valeur numérique.

Exemple :

```
Sub SubstractionExample
    Print 4 – 3                 '1
    Print 1.23e2 – 23       '100
End Sub
```

*17.2Opérateur **

Description :

Multiplie deux valeurs numériques. La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164.

Syntaxe :

Result = Expression1 * Expression2

Paramètres :

Result : Résultat de la multiplication.

Expression1, Expression2 : Toute valeur numérique.

Exemple :

```
Sub MultiplicationExample
    Print 4 * 3                 '12
    Print 1.23e2 * 23       '2829
End Sub
```

17.3Opérateur +

Description :

Additionne deux valeurs numériques. Bien que cet opérateur fonctionne sur des variables booléennes, car représentées comme des entiers, je recommande de ne pas utiliser cette fonctionnalité. Elle s'apparente à l'opérateur OR mais le transtypage peut conduire à des problèmes, l'opération étant effectuée dans le domaine des entiers et le résultat converti. La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164.

Syntaxe :

Result = Expression1 + Expression2

Paramètres :

Result : Résultat de l'addition.

Expression1, Expression2 : Toute valeur numérique .

Exemple :

```
Sub AdditionExample
    Print 4 + 3                 '7
    Print 1.23e2 + 23       '146
End Sub
```

17.4Opérateur ^

Description :

Élève le nombre à une puissance. Soit l'équation $x = y^z$. Si z est un entier, x est alors le résultat de la multiplication de y effectuée z fois (*NdT: Cet opérateur fonctionne également avec une puissance de type réel*). La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164.

Syntaxe :

Result = Expression ^ Exponent

Paramètres :

Result : Résultat de l'élévation à la puissance.

Expression : Toute valeur numérique.

Exponent : Toute valeur numérique.

Exemple :

```
Sub ExponentiationExample
    Print 2 ^ 3             '8
    Print 2.2 ^ 2       '4.84
    Print 4 ^ 0.5       '2
End Sub
```

17.5Opérateur /

Description :

Divise deux valeurs numériques. Attention, le résultat de la division peut ne pas être un entier. Utiliser la fonction Int() si nécessaire (*NdT: attention également aux divisions par zéro ;-*). La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164.

Syntaxe :

Result = Expression1 / Expression2

Paramètres :

Result : Résultat de la division.

Expression1, Expression2 : Toute valeur numérique.

Exemple :

```
Sub DivisionExample
    Print 4 /2      '2
    Print 11/2      '5.5
End Sub
```

17.6Opérateur AND

Description :

Applique un AND logique sur des valeurs booléennes et un AND bit à bit sur des valeurs numériques. Un AND bit à bit sur un type double semble impliquer une conversion en type entier. Un dépassement de capacité numérique est possible. La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164 ainsi que la table de vérité ci-dessous.

x	*y*	*x AND y*
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX
1	1	1
1	0	0
0	1	0
0	0	0

Syntaxe :

Result = Expression1 AND Expression2

Paramètres :

Result : Résultat de l'opération.

Expression1, Expression2 : Expression numérique ou booléenne.

Exemple :

```
Sub AndExample
    Print (3 And 1)              'Affiche 1
    Print (True And True)  'Affiche  -1
    Print (True And False)'Affiche  0
End Sub
```

17.7ABS (Fonction)

Description :

Retourne la valeur absolue d'une expression numérique. Si le paramètre est une chaîne de caractères, il est préalablement converti en nombre, probablement en utilisant la fonction Val(). Si le nombre est négatif, son opposé est retourné (une valeur positive donc).

Syntaxe :

Abs (Number)

Type retourné :

Double

Paramètre :

Number : Toute valeur numérique ou pouvant être évaluée comme telle.

Exemple :

```
Sub AbsExample
    Print Abs(3)        '3
    Print Abs(-4)       '4
    Print Abs("-123")  '123
End Sub
```

17.8Array (Fonction)

Description :

Crée un tableau de Variant à partir d'une liste de paramètres. C'est la méthode la plus rapide pour créer un tableau de constantes.

Attention Si vous assignez ce tableau de variant à un tableau d'un autre type, vous ne pourrez pas conserver les données si vous redimensionnez le tableau. Je pense que c'est un bogue de pouvoir assigner un tableau de Variant à un tableau d'un autre type (*NdT : l'idéal est d'utiliser les fonctions de transtypage sur les éléments comme Clng, Cdbl ...*).

Voir également la fonction DimArray.

Syntaxe :

Array (Argument list)

Type retourné :

Tableau de variants contenant la liste des arguments

Paramètre :

Argument list : Liste des valeurs séparées par des virgules.

Exemple :

```
Sub ArrayExample
    Dim a(5) As Integer
    Dim b() As Variant
    Dim c() As Integer
    Rem Array() retourne un type variant.
    Rem b est dimensionné de 0 à 9 avec b(i) = i+1
    b = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    PrintArray("b en valeur initiale", b())
    Rem b est redimensionné de 1 à 3 avec b(i) = i+1
    ReDim Preserve b(1 To 3)
    PrintArray("b après ReDim", b())
    Rem Ce qui suit n'est pas valide car le tableau "a" est déjà dimensionné à une taille différente,
    Rem mais on peut utiliser ReDim
    Rem a = Array(0, 1, 2, 3, 4, 5)

    Rem c est dimensionné de 0 à 5.
    Rem "Hello" est une chaîne de caractères mais la version 1.0.2 l'autorise
    c = Array(0, 1, 2, "Hello", 4, 5)
    PrintArray("c, Variant assigné à un tableau Integer", c())
    Rem Bizarrement, c'est permis mais c'est vide !
    ReDim Preserve c(1 To 3) As Integer
    PrintArray("c après ReDim", c())
End Sub

Sub PrintArray (lead$, a() As Variant)
    Dim i%, s$
    s$ = lead$ + Chr(13) + LBound(a()) + " à " + UBound(a()) + ":" + Chr(13)
    For i% = LBound(a()) To UBound(a())
        s$ = s$ + a(i%) + " "
    Next
    Rem J'utilise MsgBox plutôt que Print car j'ai inclus un CHR(13) (NdT: saut de ligne)
    MsgBox s$
End Sub
```

17.9ASC (Fonction)

Description :

Retourne la valeur ASCII (American Standard Code for Information Interchange) du premier caractère

de la chaîne, le reste étant ignoré. Une erreur d'exécution est générée si la chaîne est vide. Cette fonction est l'inverse de la fonction Chr.

Syntaxe :

Asc (Text As String)

Type retourné :

Integer

Paramètre :

Text: Toute chaîne de caractères non vide.

Exemple :

```
Sub AscExample
    Print Asc("ABC")        '65
End Sub
```

17.10ATN (Fonction)

Description :

Arctangente d'une expression numérique retournant une valeur comprise entre $-\pi/2$ et $\pi/2$ (radians). Cette fonction est l'inverse de la fonction tangente (Tan). Pour les non-mathématiciens, c'est une fonction trigonométrique.

Syntaxe :

ATN(Number)

Type retourné :

Double

Paramètre :

Number : Toute valeur numérique valide

Exemple :

```
Sub ExampleATN
    Dim dLeg1 As Double, dLeg2 As Double
    dLeg1 = InputBox("Entrer la longueur du côté adjacent : ","Adjacent")
    dLeg2 = InputBox("Entrer la longueur du côté opposé : ","Opposé")
    MsgBox "L'angle est " + Format(ATN(dLeg2/dLeg1), "##0.0000") _
        + " radians" + Chr(13) + "L'angle est " _
        + Format(ATN(dLeg2/dLeg1) * 180 / Pi, "##0.0000") + " degrés"
End Sub
```

17.11Beep

Description :

Génère un son système.

Syntaxe :

Beep

Exemple :

```
Sub ExampleBeep
    Beep
    Beep
End Sub
```

17.12Blue (Fonction)

Description :

Les couleurs sont représentées par un entier de type Long. Cette fonction retourne la valeur de la composante bleue de la couleur passée en argument. Voir également les fonctions RGB, Red et Green.

Syntaxe :

Blue (Color As Long)

Type retourné :

Integer compris entre 0 et 255.

Paramètre :

Color : Entier Long représentant une couleur.

Exemple :

```
Sub ExampleColor
    Dim lColor As Long
    lColor = RGB(255,10,128)
    MsgBox "La couleur " & lColor & " est composée de:" & Chr(13) &_
        "Rouge = " & Red(lColor) & Chr(13)&_
        "Vert= " & Green(lColor) & Chr(13)&_
        "Bleu= " & Blue(lColor) & Chr(13) , 64,"Couleurs"
End Sub
```

17.13ByVal (Mot-clé)

Description :

Les paramètres des procédures et des fonctions définies par l'utilisateur sont passés par référence. Si la procédure ou fonction modifie le paramètre, la modification est reportée dans la procédure ou fonction appelante. Ceci peut amener un comportement plus ou moins étrange si le paramètre appelant est une constante, ou si la procédure ou fonction appelante n'attend pas cette modification. Le mot-clé ByVal spécifie que l'argument doit être passé par valeur et non par adresse (référence).

Syntaxe :

Sub Name(ByVal ParmName As ParmType)

Exemple :

```
Sub ExampleByVal
    Dim j As Integer
    j = 10
    ModifyParam(j)
    Print j
    Rem 9
    DoNotModifyParam(j)
    Print j
    Rem toujours 9
End Sub
Sub ModifyParam(i As Integer)
    i = i - 1
End Sub
Sub DoNotModifyParam(ByVal i As Integer)
    i = i - 1
End Sub
```

Merci à Kelvin demo@onlineconnections.com.au pour ses contributions, comme avoir souligné le rôle du mot clé ByVal.

17.14Call (Instruction)

Description :

Appelle l'exécution d'une procédure, fonction ou procédure d'une DLL. Call est optionnel (sauf pour les DLL qui doivent être définies au préalable – (*NdT: Voir le mot-clé Declare*). Les arguments peuvent être écrits entre parenthèses et le doivent dans le cas d'un appel à une fonction.

Voir également : Declare

Syntaxe :

[Call] Name [(Parameters)]

Paramètres :

Name : Nom de la procédure, fonction ou procédure de DLL à appeler

Parameters : Le type et nombre de paramètres dépendent de la routine appelée.

Exemple :

```
Sub ExampleCall
    Call CallMe "Ce texte va être affiché"
End Sub
Sub CallMe(s As String)
    Print s
End Sub
```

17.15Cbool (Fonction)

Description :

Conversion du paramètre en booléen. Si l'argument est de type numérique, 0 correspond à Faux (False), toute autre valeur à Vrai (True). Si l'argument est une chaîne de caractères, “true” et “false” (indépendamment de la casse) correspondent respectivement à True et False. Tout autre valeur de la chaîne génère une erreur d'exécution. Cette fonction est utile pour forcer un résultat à être de type booléen. Si, par exemple, j'appelle une fonction qui retourne un nombre, comme InStr, je peux écrire “If CBool(InStr(s1, s2)) Then” plutôt que “If InStr(s1, s2) <> 0 Then”.

Syntaxe :

CBool (Expression)

Type retourné :

Boolean

Paramètre :

Expression : Numérique, Booléen,

Exemple :

```
Sub ExampleCBool
    Print CBool(1.334e-2)            'True
    Print CBool("TRUE")              'True
    Print CBool(-10 + 2*5)       'False
    Print CBool("John" <> "Fred")   'True
End Sub
```

17.16CByte (Fonction)

Description :

Convertit une chaîne de caractères ou une expression numérique vers le type Byte. Les chaînes de caractères sont converties en une expression numérique et les doubles sont arrondis. Si l'expression est négative ou trop grande, une erreur est générée.

Syntaxe :

Cbyte(expression)

Valeur retournée :

Byte

Paramètre :

Expression : Une chaîne de caractères ou une expression numérique

Exemple :

```
Sub ExampleCByte
    Print Int(CByte(133))              '133
    Print Int(CByte("AB"))             '0
    'Print Int(CByte(-11 + 2*5)) 'Erreur, en dehors de la plage autorisée
    Print Int(CByte(1.445532e2))   '145
    Print CByte(64.9)                      'A
    Print CByte(65)                        'A
    Print Int(CByte("12"))             '12
End Sub
```

17.17CDate (Fonction)

Description :

Convertit vers le type Date. Une expression numérique peut être interprétée comme une date, la partie entière étant le nombre de jours depuis le 31/12/1899, la partie décimale, l'heure. Les chaînes de caractères doivent être formatées comme spécifié par la convention des fonctions DateValue et TimeValue. En d'autres termes, le formatage dépend de la configuration locale de l'utilisateur. La fonction CDateFromIso n'est pas dépendante de cette configuration et pourra être utilisée le cas échéant.

Syntaxe :

CDate (Expression)

Valeur retournée :

Date

Paramètre :

Expression : Une chaîne de caractères ou expression numérique

Exemple :

```
sub ExampleCDate
    MsgBox cDate(1000.25)
    REM 26/09/1902 06:00:00
    MsgBox cDate(1001.26)
    REM 27/09/1902 06:14:24
    Print DateValue("06/08/2002")
    MsgBox cDate("06/08/2002 15:12:00")
end sub
```

17.18CDateFromIso (Fonction)

Description :

Retourne la représentation numérique d'une chaîne de caractères contenant une date au format ISO

Syntaxe :

CDateFromIso(String)

Valeur retournée :

Une date

Paramètre :

String : Une chaîne de caractères contenant une date au format ISO. L'année peut être sous deux ou quatre chiffres.

Exemple :

```
sub ExampleCDateFromIso
    MsgBox cDate(37415.70)
Rem 08/06/2002 16:48:00
    Print CDateFromIso("20020608")
    Rem 08/06/2002
    Print CDateFromIso("020608")
    Rem 08/06/1902
    Print Int(CDateFromIso("20020608"))
    Rem 37415
end sub
```

17.19CDateToIso (Fonction)

Description :

Retourne la date au format ISO à partir d'un nombre généré avec DateSerial ou DateValue.

Syntaxe :

CDateToIso(Number)

Valeur retournée :

String

Paramètre :

Number : Entier contenant la représentation numérique de la date.

Exemple :

```
Sub ExampleCDateToIso
    MsgBox "" & CDateToIso(Now) ,64,"Date ISO"
End Sub
```

17.20CDbl (Fonction)

Description :

Convertit toute expression en variable de type double. Les chaînes de caractères doivent respecter les paramétrages locaux. Sur un système paramétré en français « 12,37 » sera accepté alors que « 12.37 » générera une erreur.

Syntaxe :

CDbl (Expression)

Valeur retournée :

Double

Paramètre :

Expression : Toute chaîne de caractères ou expression numérique valide.

Exemple :

```
Sub ExampleCDbl
    Msgbox CDbl(1234.5678)
    Msgbox CDbl("1234.5678")
End Sub
```

17.21ChDir (Fonction)

Description :

Change le répertoire courant. Si vous voulez changer le lecteur courant, tapez sa lettre suivi de deux-points (D: par exemple)

Syntaxe :

ChDir (Text)

Paramètre :

Text : Une chaîne de caractères spécifiant un chemin

Exemple :

```
Sub ExampleChDir
    Dim sDir as String
    sDir = CurDir
    ChDir( "C:\temp" )
    msgbox CurDir
    ChDir( sDir )
    msgbox CurDir
End Sub
```

17.22ChDrive (Fonction)

Description :

Change le lecteur courant. La lettre du lecteur doit être en majuscules. Il est judicieux d'utiliser l'instruction OnError pour gérer toute erreur pouvant survenir (disque réseau non monté par exemple)

Syntaxe :

ChDrive (Text)

Paramètre :

Text : chaîne de caractères contenant la lettre du lecteur. La syntaxe URL est acceptée

Exemple :

```
Sub ExampleCHDrive
    On Local Error Goto NoDrive
    ChDrive "Z"
    REM Possible seulement si le disque Z existe
    Print "Fait !"
    Exit Sub
NoDrive:
    Print "Désolé, le disque n'existe pas"
    Resume Next
End Sub
```

17.23Choose (Instruction)

Description :

Retourne une valeur se situant à un endroit précis d'une liste. Si l'index excède le nombre d'éléments, la valeur Null est retournée.

Syntaxe :

Choose (Index, Selection_1[, Selection_2, ... [,Selection_n]])

Valeur retournée :

Sera du type de la valeur sélectionnée

Paramètres :

Index : Une expression numérique spécifiant la position de la valeur à retourner de la liste

Selection_i : Une valeur à retourner

Exemple :

Dans cet exemple, la variable “o” n'est pas typée donc elle prendra le type de ce qui sera sélectionné. Selection_1 est de type “String” et Selection_2 de type double. Si “o” est typée, la valeur est transformée en ce type.

```
Sub ExampleChoose
    Dim sReturn As String
    Dim sText As String
    Dim i As Integer
    Dim o
    sText = InputBox ("Entrer un nombre (1-3):","Exemple")
    i = Int(sText)
    o = Choose(i, "Un", 2.2, "Trois")
    If IsNull(o) Then
        Print "Désolé, '" + sText + "' n'est pas valide"
    Else
        Print "Obtenu '" + o + "' de type " + TypeName(o)
    End If
end Sub
```

17.24Chr (fonction)

Description :

Retourne le caractère correspondant au code ASCII ou Unicode passé en argument. On peut l'utiliser pour générer des séquences de contrôle comme les caractères d'échappement pour les imprimantes, tabulations, nouvelles lignes, retour chariots etc... On la rencontre quelquefois sous la forme “Chr$ ()”.Voir également son inverse, la fonction Asc.

Syntaxe :

Chr(Expression)

Valeur retournée :

String

Paramètre :

Expression : Variable numérique représentant une valeur valide de la table ASCII (0-255) sur 8 bits ou une valeur Unicode sur 16 bits.

Exemple :

```
Exemple :
sub ExampleChr
    REM Affiche "Ligne 1" et  "Ligne 2" sur des lignes séparées.
    MsgBox "Ligne 1" + Chr$(13) + "Ligne 2"
End Sub
```

17.25CInt (Fonction)

Description :

Convertit l'argument en un entier (Integer). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et générera une erreur.

Voir également la fonction Fix.

Syntaxe :

CInt(Expression)

Valeur retournée :

Integer

Paramètre :

Expression : chaîne ou expression numérique à convertir

Exemple :

```
Sub ExampleCInt
```

```
    Msgbox CInt(1234.5678)
    Msgbox Cint("1234,5678")
End Sub
```

17.26CLng (Fonction)

Description :

Convertit l'argument en un entier long (Long). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et générera une erreur.

Syntaxe :

CLong(Expression)

Valeur retournée :

Long

Paramètre :

Expression : chaîne ou expression numérique à convertir.

Exemple :

```
Sub ExampleCLng
    Msgbox CLng(1234.5678)
    Msgbox Clng("1234,5678")
End Sub
```

17.27Close (Instruction)

Description :

Ferme les fichiers ouverts auparavant avec l'instruction Open. Plusieurs fichiers peuvent être fermés simultanément.

Voir également Open, EOF, Kill, et FreeFile

Syntaxe :

Close #FileNumber As Integer[, #FileNumber2 As Integer[,...]]

Paramètre :

FileNumber : Entier spécifiant un fichier ouvert auparavant.

Exemple :

```
Sub ExampleCloseFile
    Dim iNum1 As Integer, iNum2 As Integer
    Dim sLine As String, sMsg As String
    'Numéro de fichier valide suivant
    iNum1 = FreeFile
    Open "c:\data1.txt" For Output As #iNum1
    iNum2 = FreeFile
    Open "c:\data2.txt" For Output As #iNum2
    Print #iNum1, "Texte dans fichier un pour le numéro " + iNum1
    Print #iNum2, "Texte dans le fichier deux pour le numéro " + iNum2
    Close #iNum1, #iNum2
    Open "c:\data1.txt" For Input As #iNum1
    iNum2 = FreeFile
    Open "c:\data2.txt" For Input As #iNum2
    sMsg = ""
    Do While not EOF(iNum1)
        Line Input #iNum1, sLine
        If sLine <> "" Then sMsg = sMsg+"Fichier: "+iNum1+":"+sLine+Chr(13)
    Loop
```

```
    Close #iNum1
    Do While not EOF(iNum2)
        Line Input #iNum2, sLine
        If sLine <> "" Then sMsg = sMsg+"Fichier: "+iNum2+":"+sLine+Chr(13)
    Loop
    Close #iNum2
    Msgbox sMsg
End Sub
```

17.28Const (Instruction)

Description :

Les constantes améliorent la lisibilité et la maintenance d'un programme en assignant un nom à une valeur et en ne donnant qu'un seul endroit dans le code pour sa définition. Les constantes peuvent être typées mais ce n'est pas obligatoire (bien que souhaitable). Une constante est définie une fois pour toute et ne peut être modifiée.

Syntaxe :

Const Text [As type] = Expression[, Text2 [As type] = Expression2[, ...]]

Paramètre :

Text : Toute chaîne de caractères respectant la convention de nommage des variables.

Exemple :

```
Sub ExampleConst
    Const iVar As String = 1964
    Const sVar = "Programme", dVar As Double = 1.00
    Msgbox iVar & " " & sVar & " " & dVar
End Sub
```

17.29ConvertFromURL (Fonction)

Description :

Conversion du nom d'un fichier au format URL en un nom de fichier dépendant du système.

Syntaxe :

ConvertFromURL(filename)

Valeur retournée :

String

Paramètre :

Filename : Chemin d'un fichier en notation URL

Exemple :

```
Sub ExampleUrl
    Dim sUrl As String, sName As String
    sName = "c:\temp\file.txt"
    sUrl = ConvertToURL(sName)
    MsgBox "Fichier original:" + sName + Chr(13) + "URL: " + sURL + Chr(13) + _
            "Et inversement:" + ConvertFromURL(sUrl)
End Sub
```

17.30ConvertToURL (Fonction)

Description :

Conversion d'un nom de fichier au format système vers un nom en notation URL

Syntaxe :

ConvertToURL(filename)

Valeur retournée :

String

Paramètre :

Filename : Nom du fichier

Exemple :

```
Sub ExampleUrl
    Dim sUrl As String, sName As String
    sName = "c:\temp\file.txt"
    sUrl = ConvertToURL(sName)
    MsgBox "Fichier original:" + sName + Chr(13) + _
           "URL: " + sURL + Chr(13) + _
           "Et inversement:" + ConvertFromURL(sUrl)
End Sub
```

17.31Cos (Fonction)

Description :

Cosinus d'une expression numérique retournant une valeur entre -1 et +1. Pour les non-mathématiciens, c'est une fonction trigonométrique.

Syntaxe :

Cos(Number)

Valeur retournée :

Double

Paramètre :

Number : Expression numérique représentant un angle en radians

Exemple :

```
Sub ExampleCos
    Dim dLeg1 As Double, dLeg2 As Double, dHyp As Double
    Dim dAngle As Double
    dLeg1 = InputBox("Entrer la longueur du côté adjacent: ","Adjacent")
    dLeg2 = InputBox("Entrer la longueur du côté opposé: ","Opposé")
    dHyp = Sqr(dLeg1 * dLeg1 + dLeg2 * dLeg2)
    dAngle = Atn(dLeg2 / dLeg1)
    MsgBox "Côté adjacent= " + dLeg1 + Chr(13) + _
           "Côté opposé = " + dLeg2 + Chr(13) + _
           "Hypothénuse = " + Format(dHyp, "##0.0000") + Chr(13) + _
           "Angle = "+Format(dAngle*180/Pi, "##0.0000")+" degrés"+Chr(13)+_
           "Cos = " + Format(dLeg1 / dHyp, "##0.0000") + Chr(13) + _
           "Cos = " + Format(Cos(dAngle), "##0.0000")
End Sub
```

17.32CreateUnoDialog (Fonction)

Description :

Crée un objet UNO représentant une boîte de dialogue UNO à l'exécution. Les boîtes de dialogue sont définies dans les bibliothèques de dialogue. Pour afficher une boîte de dialogue, elle doit être créée à partir de la bibliothèque.

Syntaxe :

CreateUnoDialog(oDlgDesc)

Valeur retournée :
Object : Boîte de dialogue à exécuter
Paramètre :
oDlgDesc : Description de la boîte de dialogue définie auparavant dans une bibliothèque (library)
Exemple :

```
Sub ExampleCreateDialog
    Dim oDlgDesc As Object, oDlgControl As Object
    DialogLibraries.LoadLibrary("Standard")
    ' Obtient la description de la boîte de dialogue dans la bibliothèque (library)
    oDlgDesc = DialogLibraries.Standard
    Dim oNames(), i%
    oNames = DialogLibraries.Standard.getElementNames()
    i = lBound( oNames() )
    while( i <= uBound( oNames() ) )
        MsgBox "Voici " + oNames(i)
        i = i + 1
    wend
    oDlgDesc = DialogLibraries.Standard.Dialog1
    ' produire un dialogue en temps réel
    oDlgControl = CreateUnoDialog( oDlgDesc )
    ' afficher le dialogue en temps réel
    oDlgControl.execute
End Sub
```

17.33CreateUnoService (Fonction)

Description :
Crée une instance d'un service UNO avec le ProcessServiceManager.
Syntaxe :
oService = CreateUnoService(UnoServiceName as string)
Valeur retournée :
??
Paramètre :
??
Exemple :

```
oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
```

17.34CreateUnoStruct (Fonction)

Description :
Crée une instance d'un type de structure UNO. Il est préférable d'utiliser la construction suivante :
Dim oStruct as new com.sun.star.beans.Property
Syntaxe :
oStruct = CreateUnoStruct(Uno type name)
Valeur retournée :

Paramètre :

Exemple :

```
    oStruct = CreateUnoStruct("com.sun.star.beans.Property")
    '**********************************
    'Voulez-vous choisir une imprimante particulière ?
    Dim mPrinter(0) As New com.sun.star.beans.PropertyValue
    mPrinter(0).Name="Nom"
    mPrinter(0).value="Autre imprimante"
    oDocument.Printer = mPrinter()
    'Vous auriez pu faire comme ci-après, après qu'il ait été défini et dimensionné
    'mPrinter(0) = CreateUnoStruct("com.sun.star.beans.PropertyValue")
```

17.35CSng (Fonction)

Description :

Convertit l'argument en un entier long (Long). Les chaînes de caractères doivent respecter le formatage local. En France, « 12.37 » ne marchera pas et générera une erreur.

Syntaxe :

CSng(Expression)

Valeur retournée :

Single

Paramètre :

Expression : chaîne ou expression numérique à convertir.

Exemple :

```
Sub ExampleCLng
    Msgbox CSng(1234.5678)
    Msgbox CSng("1234.5678")
End Sub
```

17.36CStr Function

Description :

Convertit toute expression en chaîne de caractères. Elle est généralement utilisée pour convertir les nombres en chaînes de caractères.

Input Type	*Output Value*
Boolean	“True” et “False”
Date	Chaîne avec Date et Heure
Null	Erreur d'exécution
Empty	Chaîne vide
Any Number	Le nombre en tant que chaîne. Les zéros superflus à droite de la décimale ne sont pas convertis.

Syntaxe :

CStr (Expression)

Valeur retournée :

String

Paramètre :

Expression : Toute chaîne ou valeur numérique à convertir.

Exemple :

```
Sub ExampleCSTR
Dim sVar As String
```

```
Msgbox Cdbl(1234,5678)
Msgbox Cint(1234,5678)
Msgbox Clng(1234,5678)
Msgbox Csng(1234,5678)
sVar = Cstr(1234,5678)
MsgBox sVar
end sub
```

17.37CurDir (Fonction)

Description :

Retourne le chemin courant du lecteur spécifié. Si l'argument est omis, le chemin du lecteur courant est retourné.

Syntaxe :

CurDir [(Text As String)]

Valeur retournée :

String

Paramètre :

Text : Optionnel – chaîne contenant la lettre du lecteur à analyser. Ne dépend pas de la casse.

Exemple :

```
Sub ExampleCurDir
    MsgBox CurDir("c")
    MsgBox CurDir("p")
    MsgBox CurDir()
end sub
```

17.38Date (Fonction)

Description :

Retourne ou change la date système. Le format de la date dépend de la configuration locale.

Syntaxe :

Date

Date = Text As String

Valeur retournée :

String

Paramètre :

Text : Nouvelle date système au format de la configuration locale

Exemple :

```
Sub ExampleDate
    MsgBox "La date est " & Date
End Sub
```

17.39DateSerial (Fonction)

Description :

Convertit un triplet année, mois, jour en un objet Date. La représentation interne est du type Double. La valeur 0 représente le 30 décembre 1899. On peut interpréter cette valeur comme le nombre de jours écoulés depuis cette date, les nombres négatifs représentant une date antérieure.

Voir aussi : DateValue, Date, et Day.

Attention Les années sur deux chiffres sont interprétées comme 19xx. Ceci n'est pas cohérent avec la fonction DateValue.

Syntaxe :

DateSerial (year, month, day)

Valeur retournée :

Date

Paramètres :

Year : Integer. Les valeurs entre 0 et 99 sont interprétées comme les années 1900 à 1999. Toutes les autres années doivent être spécifiées sur 4 chiffres.

Month : Integer représentant le mois. Valeurs comprises entre 1 et 12.

Day : Integer représentant le jour. Valeurs comprise entre 1 et 28, 29, 30 ou 31 (dépendant du mois).

Exemple :

```
Sub ExampleDateSerial
    Dim lDate as Long, sDate as String, lNumDays As Long
    lDate = DateSerial(2002, 6, 8)
    sDate = DateSerial(2002, 6, 8)
    MsgBox lDate
    REM retourne 37415
    MsgBox sDate
    REM retourne 08/06/2002
    lDate = DateSerial(02, 6, 8)
    sDate = DateSerial(02, 6, 8)
    MsgBox lDate
    REM retourne  890
    MsgBox sDate
    REM retourne  08/06/1902
end sub
```

17.40DateValue (Fonction)

Description :

Convertit une chaîne contenant une date en un nombre utilisable pour déterminer le nombre de jours entre deux dates.

Voir également : DateSerial, Date, et Day

Attention Les années sur deux chiffres sont considérées comme 20xx. Ceci n'est pas cohérent avec la fonction DateSerial.

Syntaxe :

DateValue [(date)]

Valeur retournée :

Long

Paramètre :

Date : String représentant une date.

Exemple :

```
Sub ExampleDateValue
    Dim s(), i%, sMsg$, l1&, l2&
    Rem Toutes ces dates correspondent au 6 Juin 2002 (NdT:au format US)
    s = Array("06.08.2002", "6.08.02", "6.08.2002", "June 08, 2002", _
        "Jun 08 02", "Jun 08, 2002", "Jun 08, 02", "06/08/2002")
```

```
    Rem Si vous utilisez ces valeurs, une erreur va être générée
    Rem ce qui contredit l'aide en ligne
    Rem s = Array("6.08, 2002", "06.08, 2002", "06,08.02", "6,08.2002", "Jun/08/02")
    sMsg = ""
    For i = LBound(s()) To UBound(s())
        sMsg = sMsg + DateValue(s(i)) + "<=" + s(i) + Chr(13)
    Next
    MsgBox sMsg
    Print "Je me suis marié il y a " + (DateValue(Date) - DateValue("6/8/2002")) + " jours"
end sub
```

17.41Day (Fonction)

Description :

Retourne le jour du mois sur la base d'une date numérique générée par DateSerial ou DateValue.

Syntaxe :

Day (Number)

Valeur retournée :

Integer

Paramètre :

Number : Date numérique telle que retournée par DateSerial

Exemple :

```
Sub ExampleDay
    Print Day(DateValue("6/8/2002")    Rem affiche 6
    Print Day(DateSerial(02,06,08))        Rem affiche 8
end sub
```

17.42Declare (Instruction)

Description :

Utilisé pour déclarer une procédure DLL (Dynamic Link Library) qui doit être exécutée à partir de OpenOffice.org Basic ; on utilisera le mot-clé ByVal si les paramètres doivent être passés par valeur et non par référence.

Voir également : FreeLibrary, Call

Syntaxe :

Declare {Sub | Function} Name Lib "Libname" [Alias "Aliasname"] [Parameter] [As Typ]

Paramètres :

Name : Un nom quelconque utilisé pour appeler la routine depuis OpenOffice.org

Aliasname : Nom de la procédure tel que défini dans la procédure.

Libname : Fichier ou nom système de la DLL. La bibliothèque est automatiquement chargée à sa première utilisation.

Parameter : Liste des paramètres, arguments à passer à la procédure lors de l'appel. (Dépendant donc de la procédure appelée).

Type : Type retourné par la fonction de la DLL. Ce type peut être omis si un « caractère de définition de type » est accolé au nom de la fonction.

Exemple :

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )
Sub ExampleDeclare
    Dim lValue As Long
    lValue = 5000
```

```
    MyMessageBeep( lValue )
    FreeLibrary("user32.dll" )
End Sub
```

17.43DefBool (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “b” soit automatiquement considérée comme une variable Booléenne.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefBool Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

```
REM Définition des types de variables par défaut
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
DefBool b-d,q
Sub ExampleDefBool
    cOK = 2.003
    zOK = 2.003
    Print cOK     Rem True
    Print zOK     Rem 2.003
End Sub
```

17.44DefDate (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “t” soit automatiquement considérée comme une variable Date.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefDate Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

```
Voir ExampleDefBool
```

17.45DefDbl (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est

spécifié. On peut ainsi permettre que toute variable commençant par “d” soit automatiquement considérée comme une variable Double.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefDbl Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

Voir ExampleDefBool

17.46DefInt (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “i” soit automatiquement considérée comme une variable Integer.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefInt Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

Voir ExampleDefBool

17.47DefLng (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “l” soit automatiquement considérée comme une variable Long.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefLng Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

Voir ExampleDefBool

17.48DefObj (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “o” soit automatiquement considérée comme une variable Object.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefObj Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

```
Voir ExampleDefBool
```

17.49DefVar (Instruction)

Description :

Définit le type par défaut des variables en accord avec une plage de caractères si aucun type n'est spécifié. On peut ainsi permettre que toute variable commençant par “v” soit automatiquement considérée comme une variable Variant.

Voir également : DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Syntaxe :

DefVar Characterrange1[, Characterrange2[, ...]]

Paramètre :

Characterrange : lettres spécifiant la plage de caractères.

Exemple :

```
Voir ExampleDefBool
```

17.50Dim (Instruction)

Description :

Déclare les variables. Le type de chaque variable est à déclarer séparément et le type par défaut est le Variant.

L'exemple suivant déclare a, b et c comme des Variant, d comme une Date.

```
Dim a, b, c, d As Date
```

Un type de variable peut également être défini en utilisant un caractère réservé ajouté à la suite du nom comme mentionné dans la section dédiée au type des variables.

Dim est utilisée pour déclarer des variables locales à l'intérieur des procédures. Les variables globales en dehors des procédures sont déclarées avec les instructions PRIVATE et PUBLIC.

A moins que l'instruction “Option Explicit” ne soit présente, les variables (en dehors des tableaux) peuvent être utilisées sans déclaration et leur type par défaut sera Variant ou cohérent avec les instructions DefBool, DefDate, DefDbL, DefInt, DefLng, DefObj et DefVar.

Les tableaux mono et multi-dimensionnels sont supportés.

Voir également : Public, Private, ReDim

Syntaxe :

[ReDim]Dim Name_1 [(start To end)] [As Type][, Name_2 [(start To end)] [As Type][,...]]

Paramètres :

Name_i : Nom de la variable ou du tableau.

Start, End : Entier constant compris entre -32768 et 32767 donnant les bornes du tableau. Au niveau de la procédure, l'instruction Redim permet des expressions numériques et des variables permettant de réinitialiser le tableau durant l'exécution.

Type : Mot-clé définissant le type de la variable. Les types supportés sont Boolean, Currency, Date, Double, Integer, Long, Object, Single, String, et Variant.

Exemple :

```
Sub ExampleDim
    Dim s1 As String, i1 As Integer, i2%
    Dim a1(5) As String
    Rem 0 à 6
    Dim a2(3, -1 To 1) As String
    Rem (0 à 3, -1 à 1)
    Const sDim as String = " Dimension:"
```

```
    For i1 = LBound(a2(), 1) To UBound(a2(), 1)
        For i2 = LBound(a2(), 2) To UBound(a2(), 2)
            a2(i1, i2) = Str(i1) & ":" & Str(i2)
        Next
    Next
    For i1 = LBound(a2(), 1) To UBound(a2(), 1)
        For i2 = LBound(a2(), 2) To UBound(a2(), 2)
            Print a2(i1, i2)
        Next
    Next
End Sub
```

17.51DimArray (Fonction)

Description :

Création d'un tableau de variant. Fonctionne comme l'instruction Dim. S'il n'y a pas d'arguments, un tableau vide est créé. Si des paramètres sont donnés, une dimension est créée pour chacun d'eux.

Voir : Array

Syntaxe :

DimArray (Argument list)

Valeur retournée :

Variant array

Paramètre :

Argument list : Optionnel. Liste d'entiers séparés par des virgules.

Exemple :

```
DimArray( 2, 2, 4 ) 'identique à DIM a( 2, 2, 4 )
```

17.52Dir (Fonction)

Description :

Retourne le nom d'un fichier, d'un répertoire ou de tous les fichiers et répertoires d'un disque qui correspondent au critère de recherche. Il est notamment possible de vérifier l'existence d'un fichier ou répertoire spécifique ou de lister les fichiers et sous-répertoires d'un répertoire donné. Si aucun fichier ou répertoire ne correspond au critère, une chaîne vide est retournée.

On appelle la fonction Dir en boucle pour itérer toutes les valeurs correspondant à ce critère jusqu'à l'obtention d'un chaîne vide.

Les attributs de répertoire et volume dans la recherche sont exclusifs : c'est la seule information qui sera retournée. Je ne peux pas déterminer lequel a la préséance car l'attribut de volume ne fonctionne pas sur la version 1.0.3.

Les attributs sont une partie de ceux disponibles avec la fonction GetAttr.

Voir également : GetAttr

Syntaxe :

Dir [(Text As String[, Attrib As Integer])]

Valeur retournée :

String

Paramètres :

Text : String spécifiant le chemin à explorer, répertoire ou fichier. La notation URL est acceptée.

Attrib : Integer Valeur sommée des attributs possibles. La fonction Dir ne retourne que les éléments répondants aux critères. Additionner les attributs pour les combiner.

Attributs	*Description*
0	Normal.
2	Fichier caché.
4	Fichier système.
8	Nom du volume (Exclusivement).
16	Répertoires (Exclusivement).

Exemple :

```
Sub ExampleDir
    REM Affiche tous les fichiers et répertoires
    Dim sFile as String, sPath As String
    Dim sDir as String, sValue as String
    Dim iFIle as Integer
    sFile= "Fichiers : "
    sDir="Répertoires :"
    iFile = 0
    sPath = CurDir
    Rem  0 : Normal.
    Rem  2 : Fichiers cachés.
    Rem  4 : Fichiers système.
    Rem  8 : Nom de volume.
    Rem 16 : Nom du répertoire uniquement.
    Rem Cet exemple ne va lister que les répertoires puisque la valeur 16 est incluse
    Rem Enlever le 16 pour lister les fichiers
    sValue = Dir$(sPath, 0 + 2 + 4 + 16)
    Do
        If sValue <> "." and sValue <> ".." Then
            If (GetAttr( sPath + getPathSeparator() + sValue) AND 16) > 0 Then
                REM Les répertoires
                sDir = sDir & chr(13) & sValue
            Else
                REM Les fichiers
                If iFile Mod 3 = 0 Then sFile = sFile + Chr(13)
                iFile = iFile + 1
                sFile = sFile + sValue &"; "
            End If
        End If
        sValue = Dir$
    Loop Until sValue = ""
    MsgBox sDir,0,sPath
    MsgBox "" & iFile & " " & sFile,0,sPath
End Sub
```

Truc	La méthode getPathSeparator() est disponible bien qu'elle n'apparaisse pas dans l'aide.
Attention	Certains OS incluent les répertoires “.” et “..” qui se réfèrent respectivement au répertoire courant et père. Si vous écrivez une macro qui traverse les répertoires vous devrez les filtrer pour éviter une boucle infinie.

Attention	Quand vous obtenez une liste de répertoires, les fichiers ne sont jamais retournés contrairement à ce qui est indiqué dans l'aide en ligne.

17.53Do...Loop (Instruction)

Description :
Instruction répétitive
Voir également : Contrôle de boucles Page 156.
Syntaxe :
Do [{While | Until} condition = True]
Bloc d'instructions
[Exit Do]
Bloc d'instructions
Loop
Syntaxe :
Do
Bloc d'instructions
[Exit Do]
Bloc d'instructions
Loop [{While | Until} condition = True]

17.54End (Instruction)

Description :
Marque la fin d'une procédure ou d'un bloc d'instructions
Voir également : Exit
Syntaxe :

Forme	*Rôle*
End	Fin d'exécution du programme. Peut être appelé à tout moment. Optionnel.
End Function	Fin d'une fonction
End If	Fin d'un bloc conditionnel If...Then...Else
End Select	Fin d'un bloc conditionnel Select Case
End Sub	Fin d'une procédure

Exemple :

```
Sub ExampleEnd
    Dim s As String
    s = InputBox ("Entrer un entier :","Testeur d'espace blanc")
    If IsWhiteSpace(Val(s)) Then
        Print "ASCII " + s + " est un espace blanc"
    Else
        Print "ASCII " + s + " n'est pas un  espace blanc"
    End If
    End
    Print "On ne passe jamais ici"
End Sub
Function IsWhiteSpace(iChar As Integer) As Boolean
```

```
    Select Case iChar
    Case 9, 10, 13, 32, 160
        IsWhiteSpace = True
    Case Else
        IsWhiteSpace = False
    End Select
End Function
```

17.55Environ (Fonction)

Description :

Retourne la valeur d'une variable d'environnement (dépendante du système d'exploitation). Sur Macintosh cette fonction retourne une chaîne vide.

Syntaxe :

Environ (Environment As String)

Valeur retournée :

String

Paramètre :

Environment : Variable d'environnement à récupérer.

Exemple :

```
Sub ExampleEnviron
    MsgBox "Path (chemin) = " & Environ("PATH")
End Sub
```

17.56EOF (Fonction)

Description :

Signale la fin d'un fichier. Permet de lire un fichier sans en dépasser la fin. Quand la fin du fichier est atteinte, EOF retourne la valeur True.

Voir également : Open, Close, Kill, et FreeFile

Syntaxe :

EOF (intexpression As Integer)

Valeur retournée :

Boolean

Paramètre:

Intexpression : Integer Valeur numérique représentant le numéro du fichier (voir Open)

Exemple :

```
Rem Exemple modifié de l'aide en ligne qui ne fonctionne pas
Sub ExampleEof
    Dim iNumber As Integer
    Dim aFile As String
    Dim sMsg as String, sLine As String
    aFile = "c:\DeleteMe.txt"
    iNumber = Freefile
    Open aFile For Output As #iNumber
    Print #iNumber, "Première ligne de texte"
    Print #iNumber, "Une autre ligne de texte"
    Close #iNumber
    iNumber = Freefile
    Open aFile For Input As iNumber
```

```
    While Not Eof(iNumber)
        Line Input #iNumber, sLine
        If sLine <>"" Then
            sMsg = sMsg & sLine & chr(13)
        End If
    Wend
    Close #iNumber
    Msgbox sMsg
End Sub
```

17.57EqualUnoObjects (Fonction)

Description :

Teste si deux objets UNO représentent la même instance d'un objet UNO.

Syntaxe :

EqualUnoObjects(oObj1, oObj2)

Valeur retournée :

Boolean

Exemple :

```
Sub ExampleEqualUnoObjects
    Dim oIntrospection, oIntro2, Struct2
    Rem Copie d'objets = même instance
    oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
    oIntro2 = oIntrospection
    print EqualUnoObjects( oIntrospection, oIntro2 )
    Rem Copie de structures : Nouvelle instance
    Dim Struct1 as new com.sun.star.beans.Property
    Struct2 = Struct1
    print EqualUnoObjects( Struct1, Struct2 )
End Sub
```

17.58EQV (opérateur)

Description :

Calcule l'équivalence logique de deux expressions. Dans une comparaison bit à bit, l'opérateur met à jour le bit dans le résultat si le bit correspondant existe dans les deux expressions ou n'existe pas dans les deux expressions. La préséance mathématique usuelle des opérateurs est applicable comme décrite page 164 ainsi que dans la table de vérité ci dessous.

x	***y***	***x EQV y***
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	VRAI
1	1	1
1	0	0
0	1	0
0	0	1

Syntaxe :

Result = Expression1 EQV Expression2

Paramètres :

Result : Variable numérique contenant le résultat de l'opération

Expression1, expression2 : Expressions à comparer

Exemple :

```
Sub ExampleEQV
    Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
    Dim vOut as Variant
    vA = 10: vB = 8: vC = 6: vD = Null
    vOut = vA > vB  EQV vB > vC
    REM retourne  -1
    Print vOut
    vOut = vB > vA  EQV vB > vC
    REM retourne -1
    Print vOut
    vOut = vA > vB  EQV vB > vD
    REM  retourne 0
    Print vOut
    vOut = (vB > vD EQV vB > vA)
    REM  retourne -1
    Print vOut
    vOut = vB EQV vA
    REM  retourne -1
End Sub
```

17.59Erl (Fonction)

Description :

Retourne le numéro de ligne à laquelle est apparue une erreur durant l'exécution.

Voir également : Err

Syntaxe :

Erl

Valeur retournée :

Integer

Exemple :

```
Sub ExampleErl
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    Rem Erreur 11 : Division par zéro
    Rem à la ligne : 8
    Rem ....
    MsgBox "Erreur " & err & ": " & error$ + chr(13) + _
        "A la ligne : " + Erl + chr(13) + Now , 16 ,"Une erreur a été générée"
End Sub
```

17.60Err (Fonction)

Description :

Retourne le numéro de la dernière erreur

Voir également : Erl

Syntaxe :

Err

Valeur retournée :

Integer

Exemple :

```
Sub ExampleErr
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    Rem Erreur 11 : Division par zéro
    Rem à la ligne : 8
    Rem ....
    MsgBox "Erreur " & err & ": " & error$ + chr(13) + _
        "A la ligne : " + Erl + chr(13) + Now , 16 ,"Une erreur a été générée"
End Sub
```

Error (Fonction)

Cette fonction est censée simuler l'apparition d'une erreur.

Cependant Error errornumber As Integer ne semble pas fonctionner.

17.61 Error (Fonction)

Description :

Retourne le message d'erreur correspondant à un code donné.

Syntaxe :

Error (Expression)

Valeur retournée :

String

Paramètre :

Expression : Optionnel. Integer contenant le code d'erreur. Si omis, le dernier message est retourné.

Exemple :

```
Sub ExampleError
    On Error GoTo ErrorHandler
    Dim iVar as Integer
    iVar = 0
    iVar = 4 / iVar
    Exit Sub
ErrorHandler:
    REM Erreur 11 : Division par zéro
    REM à la ligne : 8
    REM ....
    MsgBox "Erreur " & err & ": " & error$ + chr(13) + _
        "A la ligne : " + Erl + chr(13) + Now , 16 ,"Une erreur a été générée"
```

```
End Sub
```

Cette fonction est également censée simuler l'apparition d'une erreur.
Cependant Error errornumber As Integer ne semble pas fonctionner.

17.62Exit (Instruction)

Description :

Utilisé pour quitter les instructions de boucles, fonctions et procédures. En d'autres termes, on sort immédiatement de telles instructions. Si je suis à l'intérieur d'une procédure et que je détermine que les arguments ne sont pas corrects, je peux sortir de la procédure immédiatement.

Voir également : End

Syntaxe :

Forme	*Rôle*
Exit Do	Sort de la boucle Do...Loop courante.
Exit For	Sort de la boucle For...Next courante .
Exit Function	Sort de la fonction et poursuit l'exécution de l'appelant.
Exit Sub	Sort de la procédure et poursuit l'exécution de l'appelant.

Exemple :

```
Sub ExampleExit
    Dim sReturn As String
    Dim sListArray(10) as String
    Dim siStep as Single
    REM Construit le tableau ("B", "C", ..., "L")
    For siStep = 0 To 10
    REM Remplit le tableau avec des données de Test
        sListArray(siStep) = chr(siStep + 66)
    Next siStep
    sReturn = LinSearch(sListArray(), "M")
    Print sReturn
    Exit Sub
    REM Instruction inutile !
End Sub
REM Retourne l'index de l'entrée ou (LBound(sList()) - 1) si non trouvé
Function LinSearch( sList(), sItem As String ) as integer
    Dim iCount As Integer
    REM LinSearch recherche l'index de sItem dans sList()
    For iCount=LBound(sList()) To UBound(sList())
        If sList(iCount) = sItem Then
            LinSearch = iCount
            Exit Function
            REM Un bon usage de l'instruction Exit ici !
        End If
    Next
    LinSearch = LBound(sList()) - 1
End Function
```

17.63Exp (Fonction)

Description :

Retourne la base du logarithme népérien (e = 2.718282) élevé à la puissance de l'argument. Autrement dit, la fonction exponentielle.

Voir également : Log

Syntaxe :

Exp (Number)

Valeur retournée :

Double

Paramètre :

Number : Toute expression numérique.

Exemple :

```
Sub ExampleExp
    Dim d as Double, e As Double
    e = Exp(1)
    Print "e = " & e
    Print "ln(e) = " & Log(e)
    Print "2*3 = " & Exp(Log(2.0) + Log(3.0))
    Print "2^3 = " & Exp(Log(2.0) * 3.0)
end sub
```

17.64FileAttr (Fonction)

Description :

La première méthode d'utilisation de cette fonction sert à déterminer le mode d'accès d'un fichier utilisé lors de la commande Open. Mettre le deuxième paramètre à 1 retourne cette indication.

Valeur retournée	Mode d'accès
1	INPUT
2	OUTPUT
4	RANDOM
8	APPEND
32	BINARY

La deuxième méthode d'utilisation sert à déterminer les attributs d'un fichier MS-DOS ouvert avec l'instruction Open. Cette valeur dépend du système d'exploitation. Mettre le deuxième paramètre à 2 retourne cette indication.

Attention L'attribut d'un fichier dépend du système d'exploitation. Il est impossible d'utiliser cette fonction sur une version 32 bits pour déterminer l'attribut MS-DOS. La valeur 0 est retournée.

Voir également : Open

Syntaxe :

FileAttr (FileNumber As Integer, Attribut As Integer)

Valeur retournée :

Integer

Paramètres :

FileNumber : Numéro du fichier tel qu'utilisé pour l'ouverture par Open.

Attribut : Integer indiquant quelle information retourner. 1 = le mode d'accès, 2 = l'attribut d'accès du système de fichier.

Exemple :

```
Sub ExampleFileAttr
    Dim iNumber As Integer
    iNumber = Freefile
    Open "file:///c|/data.txt" For Output As #iNumber
    Print #iNumber, "Texte aléatoire"
    MsgBox AccessModes(FileAttr(#iNumber, 1 )),0,"Mode d'accès"
    MsgBox FileAttr(#iNumber, 2 ),0,"Attribut du fichier"
    Close #iNumber
End Sub
Function AccessModes(x As Integer) As String
    Dim s As String
    s = ""
    If (x AND 1) <> 0 Then s = "INPUT"
    If (x AND 2) <> 0 Then s = "OUTPUT"
    If (x AND 4) <> 0 Then s = s & " RANDOM"
    If (x AND 8) <> 0 Then s = s & " APPEND"
    If (x AND 32) <> 0 Then s = s & " BINARY"
    AccessModes = s
End Function
```

17.65FileCopy (Instruction)

Description :

Copie un fichier. Ne peut pas copier un fichier ouvert.

Syntaxe :

FileCopy TextFrom As String, TextTo As String

Paramètres :

TextFrom : String chemin du fichier source

TextTo : String chemin du fichier destination

Exemple :

```
Sub ExampleFilecopy
    Filecopy "c:\Data.txt", "c:\Temp\Data.sav"
End Sub
```

17.66FileDateTime (Fonction)

Description :

Retourne une chaîne de caractères avec la date de création ou de dernière modification. Le format de cette information dépend de la configuration locale, "DD/MM/YYYY HH:MM:SS" sur un ordinateur configuré en français. On utilisera ce résultat avec la fonction DateValue.

Syntaxe :

FileDateTime(Text As String)

Valeur retournée :

String

Paramètre :

Text : Nom du fichier (Jokerisation interdite). La notation URL est acceptée.

Exemple :

```
Sub ExampleFileDateTime
    REM 23/04/2003 19:30:03
    MsgBox FileDateTime("file://localhost/C|/macro.txt")
End Sub
```

17.67FileExists (Fonction)

Description :

Détermine si un fichier ou un répertoire existe.

Syntaxe :

FileExists(FileName As String | DirectoryName As String)

Valeur retournée :

Boolean

Paramètre :

FileName | DirectoryName : Fichier ou répertoire à rechercher (Jokerisation interdite)

Exemple :

```
Sub ExampleFileExists
    MsgBox FileExists("C:\autoexec.bat")
    MsgBox FileExists("file://localhost/c|/macro.txt")
    MsgBox FileExists("file:///d|/private")
End Sub
```

17.68FileLen (Fonction)

Description :

Retourne la taille d'un fichier. Si le fichier est ouvert, la taille d'avant son ouverture est retournée. On utilisera la fonction LOF pour déterminer la taille courante d'un fichier ouvert.

Syntaxe :

FileExists(FileName As String)

Valeur retournée :

Long

Paramètre :

FileName : Nom du fichier (Jokerisation interdite).

Exemple :

```
Sub ExampleFileExists
    MsgBox FileLen("C:\autoexec.bat")
    MsgBox FileLen("file://localhost/c|/macro.txt")
End Sub
```

17.69FindObject (Fonction)

Résumé :

Donnez un nom de variable et elle retournera une référence à l'objet. Voir FindPropertyObject. L'exécution du code montré ci-dessous démontre que ceci ne fonctionne pas très bien.

```
Sub TestTheThing
 Dim oTst As Object
 Dim oDoc As Object
 oTst = FindObject("oDoc")

 REM oui
```

```
 If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"

 oDoc = ThisComponent
 oTst = FindObject("oDoc")
 REM non
 If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"
 REM non
 If oTst IS ThisComponent Then Print "oTst et ThisComponent sont les mêmes"
 REM oui
 If oDoc IS ThisComponent Then Print "oDoc et ThisComponent sont les mêmes"


 oDoc = ThisComponent
 oTst = FindObject("ThisComponent")
 REM oui
 If oTst IS oDoc Then Print "oTst et oDoc sont les mêmes"
 REM oui
 If oTst IS ThisComponent Then Print "oTst et ThisComponent sont les mêmes"
 REM oui
 If oDoc IS ThisComponent Then Print "oDoc et ThisComponent sont les mêmes"


 REM ceci montre ThisComponent
 RunSimpleObjectBrowser(oTst)
 oDoc = ThisComponent
 oTst = ThisComponent.DocumentInfo
 oTst = FindPropertyObject(oDoc, "DocumentInfo")
 If IsNull(oTst) Then Print "est Null (vide)"
 If oTst IS ThisComponent.DocumentInfo Then Print "Ils sont identiques"
'
End Sub
```

17.70FindPropertyObject (Fonction)

Résumé :

Maintenant j'en ai une idée et bon sang, ceci est étrange. En plus, cela ne fonctionne pas très bien. En bref, considérez que c'est inutilisable.

Un objet contient des objets de données. Par exemple, une feuille de tableur a une propriété nommée DrawPages que je peux référencer directement avec la commande ThisComponent.DrawPages. Je peux utiliser FindPropertyObject pour obtenir une référence à cet objet.

```
obj = FindPropertyObject(ThisComponent, "DrawPages")
```

Je peux maintenant accéder à l'objet DrawPages avec la variable obj. J'ai découvert que ceci était bogué !

Exemple :

```
Sub StrangeThingsInStarBasic

Dim oSBObj1 As Object
Dim oSBObj2 As Object
Dim oSBObj3 As Object

 Set oSBObj1 = Tools
 RunSimpleObjectBrowser(oSBObj1)
 REM Nous avons également une propriété Name!!
```

```
  print oSBObj1.Name
REM  affiche @SBRTL. ?? qu'est-ce que c'est ?

  REM à propos...
  Set oSBObj2 = FindObject("Gimmicks")
  print oSBObj2.Name
REM  affiche @SBRTL de nouveau...

  REM Vous pouvez changer ce nom de propriété, mais cela est sans effet
REM   oSBObj2.Name = "Ciao" : print oSBObj2.Name
REM   oSBObj2.Name = "@SBRTL" : print oSBObj2.Name

  REM nécessite le chargement de la bibliothèque Gimmicks library maintenant
  GlobalScope.BasicLibraries.LoadLibrary("Gimmicks")

  REM l'autre vieille méthode, désapprouvée, non documentée, presque boguée ....

  REM userfields est un module dans Gimmicks library
  Set oSBObj3 = FindPropertyObject(oSBObj2, "Userfields")
  print (oSBObj3 Is Gimmicks.Userfields)

  REM nécessite le chargement de la bibliothèque Gimmicks library
  GlobalScope.BasicLibraries.LoadLibrary("Gimmicks")

  REM la fonction StartChangesUserfields est dans le the module Userfields
  REM un appel pleinement qualifié
  oSBObj2.Userfields.StartChangesUserfields
End Sub
```

17.71Fix (Fonction)

Description :

Retourne la partie entière d'une expression numérique en en retirant la partie décimale.

Voir également : CInt

Syntaxe :

Fix(Expression)

Valeur retournée :

Double

Paramètre :

Expression : Expression numérique.

Exemple :

```
sub ExampleFix
      Print Fix(3.14159)
      REM retourne 3.
      Print Fix(0)
      REM retourne  0.
      Print Fix(-3.14159)
      REM retourne  -3.
End Sub
```

17.72For...Next (Instruction)

Description :

Instruction répétitive avec un indice auto-incrémenté.

Voir également : For....Next à la Page 155.

Syntaxe :

For counter=start To end [Step step]

Bloc d'instructions

[Exit For]

Bloc d'instructions

Next [counter]

17.73Format (Fonction)

Description :

Convertit un nombre en chaîne de caractères en appliquant une mise en forme. Plusieurs mises en formes, séparées par des virgules, peuvent être spécifiées. La première sera utilisée pour les nombres positifs, la seconde pour les négatifs, la troisième pour zéro. S'il n'y a qu'un formatage, il s'applique à tous les nombres.

Code	*Description*
0	Si un nombre à un chiffre à la position du 0, ce chiffre est affiché. Cela implique que les zéros non significatifs sont affichés. Les décimales supplémentaires sont arrondies.
#	Comme 0 mais les zéros non significatifs ne sont pas affichés.
.	L'emplacement du point détermine le nombre de chiffres à placer avant et après le séparateur décimal.
%	Multiplie le nombre par 100 et affiche le signe % à l'endroit indiqué dans la chaîne de formatage.
E- E+ e- e+	Si le format contient au moins un caractère de formatage (0 ou #) à droite du symbole, le nombre est affiché en notation scientifique. La lettre E (ou e) est insérée entre le nombre et son exposant. Le nombre de caractères de formatage à droite du symbole détermine le nombre de chiffres de l'exposant. Si l'exposant est négatif, le signe moins est affiché juste avant la valeur de l'exposant. Si l'exposant est positif, le signe n'est affiché que si explicitement écrit dans le format (E+ ou e+).
,	La virgule est le caractère représentant le séparateur de milliers. Il sépare les milliers des centaines et des unités. Ce séparateur n'est affiché que s'il est encadré par des caractères de formatage de chiffres (0 or #).
- + $ ( ) espace	Plus (+), moins (-), dollar ($), espaces, ou parenthèses rentrés dans la chaîne de formatage ne sont pas interprétés, et sont donc affichés tels quels.
\	L'anti-slash est le caractère d'échappement. Il permet de ne pas interpréter le caractère suivant comme un caractère de formatage mais de l'afficher tel quel. L'anti-slash n'est pas affiché à moins de le doubler (\\). Les caractères devant être précédés d'un anti-slash pour ne pas être interprétés sont les caractères de formatage de date et heure (a, c, d, h, m, n, p, q, s, t, w, y, /, :), de numérique (#, 0, %, E, e, virgule, point), de chaîne de caractères (@, &, <, >, !). On peut également encadrer les caractères entre guillemets.
General Number	Les nombres sont affichés tels quels.

Code	*Description*
Currency	Le nombre est affiché au format monétaire en accord avec la configuration locale du poste.
Fixed	Au moins un chiffre est affiché devant le séparateur décimal.
Standard	Le nombre est affiché au format décimal en accord avec la configuration locale du poste.
Scientific	Le nombre est affiché au format scientifique.

Attention	Aucun formatage n'est effectué si le paramètre n'est pas un nombre. Une chaîne vide est retournée.
Attention	Dans la version 1.0.3.1, Format(123.555, ".##") retourne ".12356" ce qui peut être considéré comme un bogue. Changer le format en "#.##" corrige le problème. La notation « Scientific » ne fonctionne pas correctement. La notation « Currency » ne peut pas utiliser les caractères d'échappement et comporte des erreurs de formatage (US : place le symbole monétaire à droite).

Syntaxe :

Format (Number [, Format As String])

Valeur retournée :

String

Paramètres :

Number : Expression numérique à formater.

Format : Format désiré. Si omis, la fonction se comporte comme la fonction Str.

Exemple :

```
Sub ExampleFormat


REM Dépend de la configuration locale


    MsgBox Format(6328.2, "##,##0.00")
    REM = 6 328,20
    MsgBox Format(123456789.5555, "##,##0.00")
    REM = 123 456 789,56
    MsgBox Format(0.555, ".##")
    REM ,56
    MsgBox Format(123.555, "#.##")
    REM 123,56
    MsgBox Format(0.555, "0.##")
    REM 0,56
    MsgBox Format(0.1255555, "%#.##")
    REM %12,56
    MsgBox Format(123.45678, "##E-####")
    REM 12E1
    MsgBox Format(.0012345678, "0.0E-####")
    REM 1,2E3 (dysfonctionnemment)
    MsgBox Format(123.45678, "#.e-###")
    REM 1,e2
    MsgBox Format(.0012345678, "#.e-###")
```

```
    REM 1,e3 (dysfonctionnement)
    MsgBox Format(123.456789, "#.## est ###")
    REM 123.45 donne 679 (étrange)
    MsgBox Format(8123.456789, "General Number")
    REM 8123,456789
    MsgBox Format(8123.456789, "Fixed")
    REM 8 123,46
    MsgBox Format(8123.456789, "Currency")
    REM 8 123.46 € (dysfonctionnement US)
    MsgBox Format(8123.456789, "Standard")
    REM 8 123.46
    MsgBox Format(8123.456789, "Scientific")
    REM 8,12E03
    MsgBox Format(0.00123456789, "Scientific")
    REM 1,23E03 (dysfonctionnement)
End Sub
```

17.74FreeFile (Fonction)

Description :

Retourne l'index de fichier disponible pour l'ouverture d'un fichier. Assure que cet index est bien disponible et pas utilisé par un autre fichier.

Voir également : Open, EOF, Kill, et Close.

Syntaxe :

FreeFile

Valeur retournée :

Integer

Exemple :

Voir l'exemple de la fonction Close.

17.75FreeLibrary (Fonction)

Description :

Libère les ressources d'une DLL chargée par l'instruction Declare. La DLL sera automatiquement rechargée si une de ses fonctions est de nouveau appelée. Seules les DLL chargées au moment de l'exécution de la macro doivent être libérées.

Voir également : Declare

Syntaxe :

FreeLibrary (LibName As String)

Paramètre :

LibName : Nom de la DLL.

Exemple :

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )
Sub ExampleDeclare
    Dim lValue As Long
    lValue = 5000
    MyMessageBeep( lValue )
    FreeLibrary("user32.dll" )
End Sub
```

17.76Function (Instruction)

Description :

Définit une fonction utilisateur, par opposition à une procédure (Sub). On peut voir une fonction comme contenant intrinsèquement une valeur (de tout type).

Voir également : Sub

Syntaxe :

Function Name[(VarName1 [As Type][, VarName2 [As Type][,...]]]) [As Type]

Bloc d'instructions

[Exit Function]

Bloc d'instructions

End Function

Valeur retournée :

Une valeur du type de la fonction.

Exemple :

```
Function IsWhiteSpace(iChar As Integer) As Boolean
    Select Case iChar
    Case 9, 10, 13, 32, 160
        IsWhiteSpace = True
    Case Else
        IsWhiteSpace = False
    End Select
End Function
```

17.77Get (Instruction)

Description :

Lit l'enregistrement d'un fichier indexé (Random) ou une séquence d'octets d'un fichier binaire. Si le paramètre de position est omis, la lecture s'effectue à partir de la position courante. Pour les fichiers ouverts en mode binaire, cette position est exprimée en octets.

Voir également : PUT

Syntaxe :

Get [#] FileNumber As Integer, [Position], Variable

Paramètres :

FileNumber : Integer Index du fichier ouvert.

Position : Pour les fichiers ouverts en mode « Random », c'est le numéro de l'enregistrement à lire.

Variable : Variable à lire. Un type Objet ne peut pas être utilisé.

Exemple :

```
REM Ne fonctionne pas !
REM La position semble ne pas être optionnelle pour Get


Sub ExampleRandomAccess2
    Dim iNumber As Integer, aFile As String
    Dim sText As Variant
    REM de type variant obligatoirement
    aFile = "c:\data1.txt"
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=5
    Seek #iNumber,1
```

```
    REM On positionne au début
    Put #iNumber,, "1234567890"
    REM On remplit la ligne avec du texte
    Put #iNumber,, "ABCDEFGHIJ"
    Put #iNumber,, "abcdefghij"
    REM Voilà à quoi ressemble le fichier !
    REM 08 00 0A 00 31 32 33 34  35 36 37 38 39 30 08 00   ....1234567890..
    REM 0A 00 41 42 43 44 45 46  47 48 49 4A 08 00 0A 00   ..ABCDEFGHIJ....
    REM 61 62 63 64 65 66 67 68  69 6A 00 00 00 00 00 00   abcdefghij


    Seek #iNumber,1
    Get #iNumber,,sText
    Print "on open:" & sText
    Close #iNumber
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=5
    Get #iNumber,,sText
    Print "réouvert: " & sText
    Put #iNumber,,"ZZZZZ"
    Get #iNumber,1,sText
    Print "un autre Get "& sText
    Get #iNumber,1,sText
    Put #iNumber,20,"Le contenu de l'enregistrement 20"
    Print Lof(#iNumber)
    Close #iNumber
End Sub
```

17.78GetAttr (Fonction)

Description :

Retourne un nombre identifiant le type d'un « fichier ». Ces attributs sont un sur-ensemble de ceux utilisés dans la fonction Dir .

Attribut	*Description*
0	Normal
1	Lecture seule (Read only)
2	Caché (Hidden)
4	Système
8	Nom de volume
16	Répertoire (Directory)
32	Bit d'archive. Le fichier a changé depuis la dernière sauvegarde.

Voir également : Dir

Attention Ne marche pas avec la version 1.0.3.1. A tester avec la version que vous utilisez.

Syntaxe :

GetAttr (Text As String)

Valeur retournée :

Integer

Paramètre :

Text : String contenant un nom de fichier non-ambigu – La notation URL est acceptée.

Exemple :

```
Sub ExampleGetAttr
    REM devrait retourner "Read-Only Hidden System Archive"
    REM retourne" Read-Only"
    Print FileAttributeString(GetAttr("C:\IO.SYS"))
    REM devrait retourner  "Archive" mais retourne "Normal"
    Print FileAttributeString(GetAttr("C:\AUTOEXEC.BAT"))
    REM "Directory"   (répertoire)
    Print FileAttributeString(GetAttr("C:\WINDOWS"))
End Sub
Function FileAttributeString(x As Integer) As String
    Dim s As String
    If (x = 0) Then
        s = "Normal"
    Else
        s = ""
        If (x AND 16) <> 0 Then s = "Directory"
        If (x AND 1) <> 0 Then s = s & " Read-Only"
        If (x AND 2) <> 0 Then s = " Hidden"
        If (x AND 4) <> 0 Then s = s & " System"
        If (x AND 8) <> 0 Then s = s & " Volume"
        If (x AND 32) <> 0 Then s = s & " Archive"
    End If
    FileAttributeString = s
End Function
```

17.79GetProcessServiceManager (Fonction)

Description :

Accède au « central Uno service manager ». Cette fonction est requise si on doit instancier un service avec CreateInstance et contenant des arguments.

Syntaxe :

oServiceManager = GetProcessServiceManager()

Valeur retournée :

Object

Exemple :

```
REM trouver un meilleur exemple contenant un appel avec argument
oServiceManager = GetProcessServiceManager()
oIntrospection = oServiceManager.createInstance("com.sun.star.beans.Introspection");
REM C'est la même chose que l'instruction suivante
oIntrospection = CreateUnoService("com.sun.star.beans.Introspection")
```

17.80GetSolarVersion (Fonction)

Description :

Retourne le numéro interne de ˝ build ˝ (compilation) de la version courante de OpenOffice.org. Vous pouvez écrire votre macro pour contourner des bugs connus des différentes versions. Malheureusement, la fonction GetSolarVersion reste souvent la même lorsque les versions changent. La version 1.0.3.1 retourne “641” et 1.1RC3 retourne ˝ 645 ˝, mais cela n'est pas assez précis. Le code

suivant retourne la version actuelle de OOo.

```
Function OOOVersion() As String
 REM Retrouve la version courante de OOo
 REM Auteur : Laurent Godard
 REM e-mail : listes.godard@laposte.net



 Dim aSettings, aConfigProvider
 Dim aParams2(0) As new com.sun.star.beans.PropertyValue
 Dim sProvider$, sAccess$
 sProvider = "com.sun.star.configuration.ConfigurationProvider"
 sAccess   = "com.sun.star.configuration.ConfigurationAccess"
 aConfigProvider = createUnoService(sProvider)
 aParams2(0).Name = "nodepath"
 aParams2(0).Value = "/org.openoffice.Setup/Product"
 aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())

 OOOVersion=aSettings.getbyname("ooSetupVersion")
End Function
```

Syntaxe :

s = GetSolarVersion()

Valeur de retour :

String

Exemple:

```
Sub ExampleGetSolarVersion
 REM pour la 1.0.3.1, ceci vaut "641"
 Print GetSolarVersion()
End Sub
```

17.81GetSystemTicks Function

Description :

Retourne le nombre de « Ticks » fourni par le système d'exploitation. Le nombre de ticks retourné sur un intervalle de temps donné dépend toujours du système d'exploitation.

Syntaxe :

GetSystemTicks()

Valeur retournée :

Long

Exemple :

Cet exemple tente de mesurer le nombre de ticks par seconde. Sur WinXP et Ooo 1.0.3.1, on calcule 1000 ticks par seconde.

```
Sub ExampleGetSystemTicks
    Dim lTick As Long, lMillisToWait As Long
    Dim lSecsToWait As Long, lTicksPerSec As Long
    lSecsToWait = 60
    lMillisToWait = lSecsToWait * 1000
    lTick = GetSystemTicks()
    wait(lMillisToWait)
    lTick = (GetSystemTicks() - lTick)
    lTicksPerSec = lTick / lSecsToWait
    MsgBox "Chaque seconde représente " & lTicksPerSec & " Ticks"
```

```
End Sub
```

17.82GlobalScope (Objet)

Description :

Les boîtes de dialogues et macros sont organisées en bibliothèques (Library). Une bibliothèque peut contenir plusieurs macros et/ou boîtes de dialogues. En Basic, le conteneur des bibliothèques est appelé “BasicLibraries” et celui des boîtes de dialogues “DialogLibraries”. Ces bibliothèques existent à la fois au niveau global de l'application et au niveau du document. Pour appeler les conteneurs de bibliothèque globaux, il faut utiliser l'objet GlobalScope.

Syntaxe :

GlobalScope

Exemple :

```
REM  Appel Dialog1 dans la bibliothèque Standard du document
oDlgDesc = DialogLibraries.Standard.Dialog1
REM  Appel Dialog2 de la bibliothèque d'application Library1
oDlgDesc = GlobalScope.DialogLibraries.Library1.Dialog2
```

17.83GoSub (Instruction)

Description :

Transfère l'exécution vers une portion de code délimitée par un label dans la même procédure ou fonction. Les instructions suivant le label sont exécutées jusqu'à rencontrer l'instruction Return. Le programme continue alors son exécution à l'instruction suivant l'appel du GoSub.

On évite généralement d'utiliser une telle instruction que l'on remplacera avantageusement par un appel de procédure ou de fonction.

Truc	GoSub provient de vieilles versions du BASIC. L'emploi de GoSub est fortement déconseillé car il induit du code peu lisible et difficile à maintenir. L'utilisation de fonctions et procédures est recommandée.

Syntaxe :

Sub/Function
    REM Instructions
    GoSub Label
    REM Instructions
    GoSub Label
    Exit Sub/Function
Label :
    REM Blocs d'instruction
    Return
End Sub/Function

Exemple :

```
Sub ExampleGoSub
    Print "Avant le gosub"
    GoSub SillyLabel
    Print "Après le gosub"
    Exit Sub
SillyLabel:
    Print "Après le label Silly"
    Return
```

```
End Sub
```

17.84GoTo (Instruction)

Description :

Transfère l'exécution vers une portion de code délimitée par un label dans la même procédure ou fonction. Le fil d'exécution principal est perdu.

On évite généralement d'utiliser une telle instruction. Son seul intérêt peut se trouver dans la gestion des erreurs.

Voir également : On error goto

Truc	GoTo provient de vielles versions du BASIC. L'emploi de GoTo est fortement déconseillé car il induit du code peu lisible et difficile à maintenir. L'utilisation de fonctions et procédures est recommandée.

Syntaxe :

Sub/Function

    REM Instructions
    GoTo Label
    REM Instructions Jamais exécutées
    Exit Sub/Function

Label :

    Bloc d'instructions

End Sub/Function

Exemple :

```
Sub ExampleGoTo
    Print "Avant le goto"
    GoTo SillyLabel
    Print "Après le goto"
    REM Jamais exécuté
    Exit Sub
    REM Jamais exécuté
SillyLabel:
    Print "Après le label Silly"
End Sub
```

17.85Green (Fonction)

Description :

Les couleurs sont représentées par un entier de type Long. Cette fonction retourne la valeur de la composante verte de la couleur passée en argument. Voir également les fonctions RGB, Red et Blue.

Syntaxe :

Green (Color As Long)

Type retourné :

Integer compris entre 0 et 255.

Paramètre :

Color : Entier Long représentant une couleur.

Exemple :

```
    Dim IColor As Long
```

```
    lColor = RGB(255,10,128)
    MsgBox "La couleur " & lColor & " est composée de:" & Chr(13) &_
        "Rouge = " & Red(lColor) & Chr(13)&_
        "Vert= " & Green(lColor) & Chr(13)&_
        "Bleu= " & Blue(lColor) & Chr(13) , 64,"Couleurs"
End Sub
```

17.86HasUnoInterfaces (Fonction)

Description :

teste si l'objet supporte une interface UNO spécifique. Retourne True si toutes les interfaces spécifiées sont supportées.

Syntaxe :

HasUnoInterfaces(oTest, Uno-Interface-Name 1 [, Uno-Interface-Name 2, ...])

Valeur retournée :

Boolean

Paramètres :

oTest : Objet UNO à tester.

Uno-Interface-Name : Liste des noms des interfaces UNO.

Exemple :

```
Sub CloseOpenDocument
    If HasUnoInterfaces(oDoc, "com.sun.star.util.XCloseable") Then
        oDoc.close(true)
    Else
        oDoc.dispose
    End If
End Sub
```

17.87Hex (Fonction)

Description :

Retourne la valeur hexadécimale d'un nombre. Si l'argument n'est pas d'un type numérique, il est converti (si possible).

Syntaxe :

Hex(Number)

Valeur retournée :

String

Paramètre :

Number : Nombre à représenter en hexadécimal. Peut être une chaîne de caractères.

Exemple :

```
Sub ExampleHex
    Dim i1%, i2%, iNum%, s$, sFormat$, sTemp$
    iNum = 0
    s = ""
    For i1=0 To 15
        For i2=0 To 15
            s = s & " " & PrependChar(Hex(iNum), "0", 2)
            iNum = iNum + 1
        Next
        s = s & Chr(13)
```

```
    Next
    MsgBox s, 64, "Table en Hexa"
    Print Hex("64")
End Sub
Function PrependChar(s$, sPrependString$, iTotLen%) As String
    If Len(s) < iTotLen Then
        PrependChar = String(iTotLen - Len(s), sPrependString) & s
    Else
        PrependChar = s
    End If
End Function
```

17.88Hour (Fonction)

Description :

Extrait l'heure d'une valeur de temps retournée par TimeSerial ou TimeValue.

Syntaxe :

Hour(Number)

Valeur retournée :

Integer

Paramètre :

Number : Expression numérique contenant une valeur de temps.

Exemple :

```
Sub ExampleHour
    Print "L'heure courante est " & Hour( Now )
    Print Hour(TimeSerial(14,08,12))
    Print Hour(TimeValue("14:08:12"))
End Sub
```

17.89If ... Then ... Else (Instruction)

Description :

Permet d'exécuter un bloc d'instructions suivant qu'une condition est évaluée à True ou False. Bien que l'on puisse utiliser les instructions GoTo ou GoSub pour sortir d'un If (!!!!!!!!), on ne peut les utiliser pour rentrer dans un bloc d'instructions contenu dans un If.

Syntaxe :

If condition=True Then
    Bloc d'instructions
[ElseIf condition=True Then]
    Bloc d'instructions
[Else]
    Bloc d'instructions
End If

Syntaxe :

If condition Then Bloc d'instructions

If (condition=False) Then Bloc d'instructions

Exemple :

```
Sub ExampleIf
    Dim i%
```

```
    i% = 4
    If i < 5 Then
        Print "i est plus petit que 5"
        If i = 4 Then Print "i est égal à  4"
        If i < 3 Then
            Print "i est plus petit que 3"
        End If
    ElseIf i = 5 Then
        Print "i est égal à 5"
    Else
        Print "i est plus grand que 5"
    End If
End Sub
```

17.90IIF (Instruction)

Description :

Retourne un résultat suivant que la condition spécifiée est évaluée à True ou False. Bien que cette commande soit très appréciable, elle semble avoir quelques petits dysfonctionnements avec la version 1.0.3.1

Syntaxe :

IIf (Expression, ExpressionTrue, ExpressionFalse)

Valeur retournée :

ExpressionTrue ou ExpressionFalse

Paramètre :

Expression : Expression conditionnelle à évaluer.

ExpressionTrue : Valeur retournée si la condition est True (Vraie)

ExpressionFalse : Valeur retournée si la condition est False (Fausse)

Exemple :

```
Sub IIfExample
    Print IIf(3>4,"Oui", "Non")
    REM Non
    Print IIf(4>2,"Oui", "Non")
    REM Oui
End Sub
```

17.91Imp (Opérateur)

Description :

Calcule l'implication logique de deux expressions

En cours de rédaction – Non traduit

Syntaxe :

Result = Expression1 Imp Expression2

Exemple :

```
Sub ExampleImp
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
A = 10: B = 8: C = 6: D = Null
vOut = A > B Imp B > C
REM retourne -1
```

```
vOut = B > A Imp B > C
REM retourne  0
vOut = A > B Imp B > D
REM retourne  -1
vOut = (B > D Imp B > A)
REM retourne  0
vOut = B Imp A
REM retourne  -3
End Sub
```

17.92Input (Instruction)

Description :

L'instruction Input est utilisée pour lire séquentiellement les données d'un fichier ouvert et les affecter à une ou plusieurs variables. Le retour chariot (Asc=13), la fin de ligne (Asc=10) et la virgule agissent comme délimiteurs. Quand une valeur numérique est lue, l'espace est également utilisé comme délimiteur. Lire une chaîne non numérique dans une variable numérique met sa valeur à 0.

Il n'est pas possible de lire les virgules et les guillemets avec cette instruction. Vous devrez alors utiliser l'instruction LineInput.

Voir également : Open, Line Input#, Close, Eof, Get

Syntaxe :

Input #FileNumber var1[, var2[, var3[,...]]]

Paramètres :

FileNumber : Indicateur de fichier utilisé lors de l'instruction Open.

var : Variables de type string ou numérique dans lesquelles mettre le contenu de ce qui est lu.

Exemple :

```
??
```

17.93InputBox (Fonction)

Description :

Affiche une demande à l'utilisateur dans une boite de dialogue. L'annulation retourne une chaîne vide. Si aucune position n'est spécifiée, la boite est centrée à l'écran.

Syntaxe :

InputBox (Msg [, Title[, Default[, x_pos, y_pos As Integer]]])

Type retourné :

String

Paramètres :

Msg : Message à afficher.

Title : Titre à afficher dans barre la fenêtre.

Default : Chaîne réponse par défaut.

x_pos : Position horizontale absolue en Twips.

y_pos : Position verticale absolue en Twips.

Exemple :

```
Sub ExampleInputBox
    Dim s$
    s = InputBox ("Message","Titre", "défaut")
```

```
    MsgBox ( s , 64, "Confirmation de la phrase")
End Sub
```

17.94InStr (Fonction)

Description :

Retourne la position d'une chaîne dans une autre. Si la chaîne n'est pas trouvée, retourne 0.

Attention Dans la version 1,1RC2, la variable retournée est de type Integer mais la valeur potentiellement retournée peut être supérieure car une String peut avoir une longueur de 64 K. Une valeur négative est alors retournée si la valeur de la position est trop grande.

```
Sub BugInStr
 Dim b$, i&
 b$ = String(40000, "a") & "|"
REM le caractère 40,001 est un "|"
 i = instr(b, "|")
REM -25535
 MsgBox cstr(i) & " ou " & (65536 + i)
REM -25535 ou 40001
End Sub
```

Syntaxe :

InStr([Start As Integer,] Text1 As String, Text2 As String[, Compare])

Type retourné :

Integer

Paramètres :

Start : Optionnel - Position du début de la recherche. Par défaut 1, début de la chaîne.

Text1 : Chaîne dans laquelle effectuer la recherche.

Text2 : Chaîne à rechercher.

Compare : Si 1, recherche indépendante de la casse, 0 (par défaut), recherche binaire.

Exemple :

```
Sub ExampleInStr
    Dim s$
    s = "SbxInteger getTruck(SbxLong)"
    RemoveFromString(s, "Sbx")
    Print s
End Sub

REM Efface toutes les occurences bad$ dans s$
REM modifie la chaîne s$
Sub RemoveFromString(s$, bad$)
    Dim i%
    i = InStr(s, bad)
    Do While i > 0
        Mid(s, i, Len(bad), "")
        i = InStr(i, s, bad)
    Loop
End Sub
```

Attention On ne peut pas utiliser l'option « Compare » si on utilise l'option « Start ».

17.95Int (Fonction)

Description :

Retourne le premier entier inférieur à l'argument. La valeur absolue de cet entier est donc plus petite pour les nombres positifs et plus grande pour les négatifs.

Voir également : CInt, Fix

Syntaxe :

Int (Number)

Type retourné :

Double

Paramètre :

Number : Toute expression numérique valide.

Exemple :

```
Sub ExampleINT
    Print " " & Int(3.14159) & " " & Fix(3.14)
    REM 3 3
    Print " " & Int(0) & " " & Fix(0)
    REM 0 0
    Print " " & Int(-3.14159) & " " & Fix(-3.1415)
    REM -4 -3
    Print " " & Int(2.8) & " " & Fix(2.8)
    REM 2 2
End Sub
```

Attention -3.4 est arrondi en -4. Utiliser Fix si on veut la partie entière.

17.96IsArray (Fonction)

Description :

Teste si une variable est un tableau.

Syntaxe :

IsArray(Var)

Valeur retournée :

boolean

Paramètre :

Var : Toute variable à tester à condition qu'elle soit déclarée en tant que tableau.

Exemple :

```
Sub ExampleIsArray
    Dim sDatf(10) as String, i
    Print IsArray(sDatf())
```

```
REM True
    Print IsArray(i())
REM False
End Sub
```

17.97 IsDate (Fonction)

Description :

Teste si un nombre ou texte peut être converti en Date.

Syntaxe :

IsDate(Expression)

Valeur de retour :

Booléen

Paramètres :

Expression : toute expression chaîne ou numérique à tester.

Exemple :

```
Sub ExampleIsDate
    Print IsDate("12.12.1997")
    REM True
    Print IsDate("12121997")
    REM False
End Sub
```

17.98 IsEmpty (Fonction)

Description :

Teste si une variable Variant contient la valeur « Empty », indiquant que la variable n'a pas été initialisée.

Voir aussi : « Object, Variant, Empty et Null » .

Syntaxe :

IsEmpty(Var)

Valeur de retour :

Booléen

Paramètre :

Var : la variable à tester

Exemple :

```
Sub ExampleIsEmpty
    Dim v1 as Variant, v2 As Variant, v3 As Variant
    v2 = Null : v3 = "hello"
    Print IsEmpty(v1)
    REM True
    Print IsEmpty(v2)
    REM False
    Print IsEmpty(v3)
    REM False
```

```
    v2 = Empty
    REM  ?? Supprimé après la version 1.0.3.1
    Print IsEmpty(v2)
    REM  Devrait renvoyer True (Vrai)
End Sub
```

17.99 IsMissing (Fonction)

Description :

Teste si une procédure ou une fonction a été appelée avec ou sans un paramètre optionnel. Le paramètre doit être déclaré avec le mot clé « Optional » pour que cela fonctionne. A partir de la version 1.0.3.1, il y a eu apparition d'erreurs mineures comme mentionné dans la section 11.3.1 sur les paramètres optionnels.

Syntaxe :

IsMissing(var)

Valeur retournée :

Booléen

Paramètre :

Var : Variable à tester

Exemple :

```
Function FindCreateNumberFormatStyle (sFormat As String, Optional doc, Optional locale)
    Dim oDocument As Object
    Dim aLocale as new com.sun.star.lang.Locale
    Dim oFormats As Object
    REM S'il n'a pas été envoyé par l'appel, alors on utilise ThisComponent
    oDocument = IIf(IsMissing(doc), ThisComponent, doc)
    oFormats = oDocument.getNumberFormats()
    ....
End Function
```

17.100IsNull (Fonction)

Description :

Teste si un Variant ou un Objet contient la valeur spéciale « Null » indiquant que la variable ne contient aucune valeur. Un Objet non initialisé est Null, un Variant non initialisé est Empty (Vide), mais il peut être initialisé et contenir la valeur Null.

Voir également : IsEmpty, macro à inclure GetSomeObjInfo

Syntaxe :

IsNull(Var)

Valeur retournée :

Booléen

Paramètre :

Var : variable à tester

Exemple :

```
Sub ExampleIsNull
    Dim v1 as Variant, v2 As Variant, v3 As Variant, o As Object
```

```
    v2 = Null : v3 = "hello"
    Print IsNull(v1)
    REM  False
    Print IsNull(v2)
    REM  True
    Print IsNull(v3)
    REM False
    v3 = Null
    Print IsNull(v3)
    REM True
    Print IsNull(o)
    REM True
End Sub
```

17.101IsNumeric (Fonction)

Description :

Teste si l'expression passée en argument est un nombre ou pourrait être convertie en nombre.

Syntaxe :

IsNumeric(Var)

Valeur retournée :

Booléen

Paramètre :

Var : toute expression à tester

Exemple

```
Sub ExempleIsNumeric
    Dim v1, v2, v3
    v1 = "abc" : v2 = "123" : v3 = 4
    Print IsNumeric(v1)
    REM  False
    Print IsNumeric(v2)
    REM  True
    Print IsNumeric(v3)
    REM  True
    Print IsNumeric("123x")
    REM  False
End Sub
```

17.102IsObject (Fonction)

Description :

Selon la documentation en ligne, cette fonction teste si l'objet transmis est un objet OLE. Après un coup d'œil au code source et quelques essais, il s'avère que cette fonction retourne également True pour tout objet régulier.

Voir aussi : Macro à inclure GetSomeObjInfo

Syntaxe :

IsObject(ObjectVar)

Valeur retournée :

Booléen

Paramètres :

ObjectVar : Toute variable à tester

Exemple :

```
Sub ExampleIsObject
    Dim o As Object, s AS String
    Print IsObject(o)
    REM  True
    Print IsObject(s)
    REM  Erreur d'exécution : objet non initialisé
End Sub
```

17.103IsUnoStruct (Fonction)

Description :

Renvoie True si l'objet transmis en paramètre est un objet UNO. L'aide en ligne indique à tort que le paramètre peut être un nom plutôt qu'un objet.

Voir aussi : macro à inclure GetSomeObjInfo.

Syntaxe :

IsUnoStruct(var)

Valeur retournée :

Booléen

Paramètres :

Var : objet à tester

Exemple :

```
Sub ExampleIsUnoStruct
    Dim o As Object, s AS String
    Dim aProperty As New com.sun.star.beans.Property
    Print IsUnoStruct(o)
    REM  False
    Print IsUnoStruct("com.sun.star.beans.Property")
    REM  False
    Print IsUnoStruct(aProperty)
    REM  True
End Sub
```

17.104Kill (Fonction)

Description :

Efface un fichier du disque. Toute notation de fichier peut être utilisée, mais les caractères génériques ne sont pas acceptés.

Syntaxe :

Kill(Nom_de_fichier)

Valeur retournée :

Aucune

Paramètres :

Nom_de_fichier : nom du fichier à effacer.

Exemple :

```
Sub ExampleKill
    Kill "C:\datafile.dat"
End Sub
```

17.105LBound (Fonction)

Description :

Renvoie l'indice de début d'un tableau. Un tableau ne commence pas obligatoirement à l'indice 0.

Syntaxe :

LBound(ArrayName [, Dimension])

Valeur retournée :

Entier (Integer)

Paramètres :

ArrayName : Nom du tableau

Dimension : entier indiquant quelle dimension est recherchée. Par défaut, la première dimension est retournée.

Exemple :

```
Sub ExampleUboundLbound
    Dim a1(10 to 20) As String, a2 (10 to 20,5 To 70) As String
    print "(" & LBound(a1()) & ", " & UBound(a1()) & ")"
    REM  (10, 20)
    print "(" & LBound(a2()) & ", " & UBound(a2()) & ")"
    REM  (10, 20)
    print "(" & LBound(a2(),1) & ", " & UBound(a2(),1) & ")"
    REM  (10, 20)
    print "(" & LBound(a2(),2) & ", " & UBound(a2(),2) & ")"
    REM  (5, 70)
End Sub
```

17.106LCase (Fonction)

Description :

Retourne la valeur de l'argument en minuscules.

Syntaxe :

LCase (String)

Type retourné :

String

Paramètre :

String : Chaîne à retourner en minuscules.

Exemple :

```
Sub ExampleLCase
    Dim s$
    s = "Las Vegas"
    Print LCase(s)
    REM  "las vegas"
    Print UCase(s)
    REM  "LAS VEGAS"
```

```
end Sub
```

17.107Left (Fonction)

Description :

Retourne les n caractères à gauche d'une chaîne.

Attention	Dans la version 1.1RC2, le paramètre de Left est un Integer alors que la chaîne ne peut être longue que de 64K.

Syntaxe :

Left(String, Integer)

Valeur retournée :

String

Paramètres :

String : Expression chaîne

Integer : Nombre de caractères à retourner. Si 0, une chaîne de longueur nulle est retournée.

Exemple :

```
Print Left("123456789", 2)
REM Affiche 12
```

17.108Len (Fonction)

Description :

Retourne le nombre de caractères (la longueur) d'une chaîne, le nombre d'octets nécessaires à stocker une variable.

Syntaxe :

Len(Text As String)

Valeur retournée :

Long

Paramètres :

Text : Expression chaîne ou une variable d'un autre type.

Exemple :

```
Sub ExampleLen
    Dim s$, i%
    s = "123456"
    i = 7
    Print Len(s)
    REM 6
    Print Len(i)
    REM 1
    Print Len(1134)
    REM 4
    Print Len(1.0/3)
    REM 17
End Sub
```

17.109Let (Mot clé)

Description :

Mot clé optionnel indiquant qu'une valeur doit être assignée à une variable (rarement utilisé).

Syntaxe :

[Let] VarName=Expression

Valeur retournée :

Aucune

Paramètres :

VarName : Variable à laquelle la valeur doit être attribuée.

Exemple :

```
Sub ExampleLet
    Dim s$
    Let s = "Las Vegas"
End Sub
```

17.110Line Input (Instruction)

Description :

Lit des chaînes de caractères depuis un fichier texte séquentiel vers une variable. Vous devez d'abord ouvrir le fichier avec l'instruction Open. Les variables sont lues ligne par ligne jusqu'au premier retour chariot (code ASCII 13) ou changement de ligne (code ASCII 10). Le caractère de fin de ligne n'est pas inclus dans la variable de lecture.

Syntaxe :

Line Input #FileNumber As Integer, Var As String

Valeur retournée :

Aucune

Paramètres :

FileNumber : Numéro du fichier ouvert depuis lequel les variables doivent être lues.

var : Variable utilisée pour stocker le résultat.

Exemple :

Voir l'exemple à la page .

17.111Loc (Fonction)

Description :

La fonction Loc retourne la position courante dans un fichier ouvert. Si elle est utilisée pour un fichier à accès direct, elle retourne le numéro du dernier enregistrement auquel on a accédé. Pour un fichier séquentiel, la fonction retourne la position dans le fichier divisée par 128. Pour un fichier binaire, la position du dernier octet lu ou écrit est retournée (À vérifier).

Syntaxe :

Loc(FileNumber)

Valeur retournée :

Long

Paramètres :

FileNumber : Expression numérique contenant le numéro d'un fichier ouvert.

Exemple :

```
??
```

17.112Lof (Fonction)

Description :

Lof retourne la taille d'un fichier en octets. Pour obtenir la longueur d'un fichier non ouvert, utiliser plutôt la fonction FileLen.

Syntaxe :

Lof(FileNumber)

Valeur retournée :

Long

Paramètres :

FileNumber : Expression numérique contenant le numéro d'un fichier ouvert.

Exemple :

À VÉRIFIER

```
Sub ExampleRandomAccess
    Dim iNumber As Integer
    Dim sText As Variant
    REM doit être un Variant
    Dim aFile As String
    aFile = "c:\data.txt"
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=32
    Seek #iNumber,1
    REM Position de départ
    Put #iNumber,, "C'est la première ligne de texte"
    REM Remplit avec du texte
    Put #iNumber,, "C'est la seconde ligne de texte"
    Put #iNumber,, "C'est la troisième ligne de texte"
    Seek #iNumber,2
    Get #iNumber,,sText
    Print sText
    Close #iNumber
    iNumber = Freefile
    Open aFile For Random As #iNumber Len=32
    Get #iNumber,2,sText
    Put #iNumber,,"C'est une nouvelle ligne de texte"
    Get #iNumber,1,sText
    Get #iNumber,2,sText
    Put #iNumber,20,"C'est le texte de l'enregistrement n° 20"
    Print Lof(#iNumber)
    Close #iNumber
End Sub
```

17.113Log (Fonction)

Description :

Retourne le logarithme naturel d'un nombre. Le logarithme naturel est le logarithme en base e, qui est une constante de valeur approximative 2,718282... Le calcul du logarithme en base n quelconque est donné en divisant le logarithme naturel du nombre par le logarithme naturel de n, par la formule Logn(x) = Log(x) / Log(n).

Syntaxe :

Log(Number)

Valeur retournée :

Double

Paramètres :

Number : Expression numérique dont on veut calculer le logarithme naturel.

Exemple :

```
Sub ExampleLogExp
    Dim a as Double
    Dim const b1=12.345e12
    Dim const b2=1.345e34
    a=Exp( Log(b1)+Log(b2) )
    MsgBox "" & a & chr(13) & (b1*b2) ,0,"Multiplication via le logarithme"
End Sub
```

17.114Loop (Instruction)

Description :

L'instruction Loop est utilisée pour répéter des instructions tant qu'une condition est True (vraie), ou jusqu'à ce qu'une instruction soit vraie. Voir le traitement de boucles Do... à la page 156.

Syntaxe :

Do [{While | Until} condition = True]

bloc d'instructions

[Exit Do]

bloc d'instructions

Loop

Syntaxe :

Do

bloc d'instructions

[Exit Do]

bloc d'instructions

Loop [{While | Until} condition = True]

Exemple :

```
Sub ExampleDoLoop
    Dim sFile As String, sPath As String
    sPath = "c:\" : sFile = Dir$( sPath ,22)
    If sFile <> "" Then
```

```
    Do
        MsgBox sFile
        sFile = Dir$
        Loop Until sFile = ""
    End If
End Sub
```

17.115LSet (Instruction)

Description :

Lset permet de justifier à gauche une chaîne de caractères à l'intérieur de l'espace utilisé par une autre chaîne. Toutes les positions restantes à gauche seront remplies par des espaces. Si la nouvelle chaîne ne tient pas dans l'ancienne, elle sera tronquée. ?? Ceci ne marche pas pour la version 1.0.3.1.

Lset permet également de remplacer des données depuis un type de données utilisateur vers un autre. Cela utilise tous les octets d'une structure de données et les remplace par les autres, en ignorant la structure sous-jacente. Ceci est actuellement d'une utilité réduite, sachant que OOo Basic ne supporte pas les types de données définis par un utilisateur.

Syntaxe :

LSet Var As String = Text

LSet Var1 = Var2

Paramètres :

Var : Toute variable de type chaîne, dans laquelle le texte doit être aligné à gauche.

Text : Le texte à aligner à gauche.

Var1 : Nom de la variable de type utilisateur destination.

Var2 : Nom de la variable de type utilisateur source.

Exemple :

```
Sub ExampleLSet
    Dim sVar As String, sExpr As String
    sVar = String(40,"*")
    sExpr = "SBX"
    REM Aligne à gauche "SBX" dans la chaîne de référence de 40 caractères de long
    LSet sVar = sExpr
    Print ">"; sVar; "<"
    REM ">SBX<" Ne marche pas, devrait contenir les espaces.
    sVar = String(5,"*")
    sExpr = "123456789"
    LSet sVar = sExpr
    Print ">"; sVar; "<"
    REM ">12345<"
End Sub
```

17.116LTrim (Fonction)

Description :

Retire tous les espaces au début d'une expression chaîne.

Syntaxe :

LTrim(Text)

Valeur retournée :

String

Paramètres :

Text : Toute expression de type chaîne

Exemple :

```
Sub ExampleSpaces
    Dim sText2 As String,sText As String,sOut As String
    sText2 = " <*Las Vegas*> "
    sOut = """+sText2 +"""+ Chr(13)
    sText = Ltrim(sText2)
    REM sText = <*Las Vegas*> "
    sOut = sOut + """+sText +"" + Chr(13)
    sText = Rtrim(sText2)
    REM sText = " <*Las Vegas*> "
    sOut = sOut +"""+ sText +"" + Chr(13)
    sText = Trim(sText2)
    REM sText = " <*Las Vegas*> "
    sOut = sOut +"""+ sText +""
    MsgBox sOut
End Sub
```

17.117Private (mot-clé)

Description :

Le mot clé Private est utilisé pour déclarer une variable comme privée au module. Si une variable est déclarée avec le mot-clé Dim, elle est considérée comme privée. Voir la section Dim pour la description de la syntaxe.

Voir également : Dim, Public

Syntaxe :

Private Name_1 [(start To end)] [As VarType][, Name_2 [(start To end)] [As VarType][,...]]

Exemple :

```
Private iPriv As Integer
Sub ExamplePublic
    iPriv = 1
    Call CalledSub
End Sub
Sub CalledSub
    Print iPriv     REM 1
End Sub
```

17.118Public (mot-clé)

Description :

Le mot clé Public est utilisé pour déclarer une variable comme accessible par tous les modules. Si une variable est déclarée avec le mot-clé Dim, elle est considérée comme privée. Voir la section Dim pour la description de la syntaxe.

Voir également : Dim, Private

Syntaxe :

Public Name_1 [(start To end)] [As VarType][, Name_2 [(start To end)] [As VarType][,...]]

Exemple :

```
Public iPub As Integer
Sub ExamplePublic
    iPub = 1
    Call CalledSub
End Sub
Sub CalledSub
    Print iPub    REM 1
End Sub
```

17.119Red (Fonction)

Description :

Les couleurs sont représentées par un entier de type Long. Cette fonction retourne la valeur de la composante rouge de la couleur passée en argument. Voir également les fonctions RGB, Blue et Green.

Syntaxe :

Red (Color As Long)

Type retourné :

Integer compris entre 0 et 255.

Paramètre :

Color : Entier Long représentant une couleur.

Exemple :

```
    Dim lColor As Long
    lColor = RGB(255,10,128)
    MsgBox "La couleur " & lColor & " est composée de:" & Chr(13) &_
        "Rouge = " & Red(lColor) & Chr(13)&_
        "Vert= " & Green(lColor) & Chr(13)&_
        "Bleu= " & Blue(lColor) & Chr(13) , 64,"Couleurs"
End Sub
```

17.120Shell Function

Description :

Lance une application externe. Le style de fenêtre de l'application démarrée peut optionnellement être paramétré avec les valeurs suivantes :

Style	*Description*
0	Focus sur une fenêtre cachée du programme.
1	Focus sur la fenêtre d'application au format standard.
2	Focus sur la fenêtre d'application minimisée.
3	Focus sur une fenêtre d'application maximisée.
4	Taille de fenêtre standard d'application, sans le focus.
6	Taille de fenêtre d'application minimisée, mais le focus reste sur la fenêtre active.
10	Affichage plein écran.

Le programme est censé démarrer et continuer à fonctionner en arrière plan sauf si le dernier paramètre (bsync) est positionné à True. Ceci signifie que le contrôle est renvoyé immédiatement depuis la commande shell.

Le type de retour n'est pas spécifié dans l'aide en ligne. Expérimentalement, j'ai déterminé que ce type est LONG. La valeur de retour a toujours été zéro lorsque j'ai pris la peine de vérifier. Si le programme n'existe pas, alors une erreur est générée et la macro s'arrête.

Syntaxe :

Shell (Pathname As String[, Windowstyle As Integer][, Param As String][, bSync])

Valeur de retour :

Long

Paramètres :

Pathname : Chemin complet et nom du programme à lancer.

Windowstyle : Spécifie le style de la fenêtre dans laquelle le programme sera lancé.

Param : N'importe quelle chaîne de caractère telle qu'elle puisse être passée en ligne de commande.

Bsync : Si False (défaut), un retour immédiat est exécuté. Si True, alors l'état du Shell ne sera retourné qu'après terminaison du programme.

Exemple :

```
Sub ExampleShell
 Dim vRC As Variant
 REM Une fenêtre de type 2  s'affichant en avant



 vRC = Shell("C:\andy\TSEProWin\g32.exe", 2, "c:\Macro.txt")
 Print "Je suis de retour, et le code de retour est " & vRC
 REM Ces deux-ci ont des espaces dans les noms
 Shell("file:///C|/Andy/My%20Documents/oo/tmp/h.bat",2)
 Shell("C:\Andy\My%20Documents\oo\tmp\h.bat",2)
End Sub
```

Antal Attila <atech@nolimits.ro> nous a transmis l'exemple suivant de l'utilisation de l'argument bsync.

```
Sub Main()
 REM  Il faut d'abord créer sur votre disque un fichier avec le contenu suivant:
 REM  sous Windows (nom de fichier C:\tmp\test.bat)
 REM       echo %1
 REM       pause
 REM  sous Linux (nom de fichier /home/guest/Test.sh)
 REM       echo $1
 REM       sleep 100000

 REM  ------------   Exemple de Sync  ------------------
 REM  appel de ma macro d'exécution de script avec bSync=TRUE
 REM  l'exécution du basic attendra que le terminal (ou la fenêtre msdos)
 REM  soit fermé par l'appui d'une touche (CTRL+C sous Windows)
 REM Sous Windows
 shellRunner("file://C:/tmp/", "Test", "Helo World", TRUE)
 REM  ou sous Linux
 shellRunner("file:///home/guest/", "Test", "Helo World", TRUE)
 REM  Signaler la fin de l'exécution
 Print "The End"

 REM  ------------   Sans Sync  ------------------
 REM  Appel avec bSync=FALSE
 REM  L'exécution du code basic sera continuée
 REM  Sous Windows
```

```
 shellRunner("file://C:/tmp/", "Test", "Helo World", FALSE)
 REM  ou sous Linux
 shellRunner("file:///home/guest/", "Test", "Helo World", FALSE)
 REM  On indique la fin de l'exécution
 Print "The End"
End Sub


Sub shellRunner(dirPath$, script$, prms$, sync as Boolean)
 Dim filePath$, ef$, ed$, isWindows as Boolean

 REM  On regarde sous quel OS on se trouve
 If instr(mid(dirPath,8),":/")>0 or instr(dirPath,8),"\")>0 Then
   isWindows=TRUE
 Else
   isWindows=FALSE
 End If

 REM  Conversion de l'URL en chemin de fichier
 filePath = convertFromURL(dirPath)

 REM  Création de la ligne de commande
 If isWindows Then
    ef = "command.com /C "+filePath+script+".bat"
 Else
    ef = "xvt -e sh "+filePath+script+".sh"
 End If

 REM  Exécution de la ligne de commande
 Shell(ef, 1, prms, sync)
End Sub
```

17.121Notation URL et Noms de fichiers

Il est conseillé de lire la description des fonctions ConvertToURL et ConvertFromURL.

17.121.1Notation URL

Sous le système d'exploitation Windows, “c:\autoexec.bat” est un exemple de nom de fichier. On peut également définir celui-ci en notation URL comme “file:///c|/autoexec.bat”.

De manière générale quand on effectue une telle conversion, on débute l'URL avec “file:///”, on change “:” en “|”, et on remplace “\” par “/”. Si on veut insérer le nom de l'ordinateur ou son adresse IP, on l'insère entre le deuxième et troisième Slash (/), comme ceci : “file:///localhost/c|/autoexec.bat/”.

17.121.2Chemins avec des espaces et autres caractères spéciaux.

Les espaces et caractères spéciaux pouvant être inclus dans une notation URL doivent l'être avec une séquence d'échappement. Prenez la valeur ASCII du caractère, convertissez-la en hexadécimal, mettez le caractère % devant et placez-là où vous désirez voir apparaître le caractère. Par exemple, pour inclure un espace, “c:\My Documents\info.sxw” devient “file///c|/My%20Documents/info.sxw”.

18Index