Introduction à KiXtart

Sommaire

1. Présentation

KiXtart est un langage de scripts fonctionnant sous Windows développé par Ruud van Veslen de Microsoft Hollande. Ce langage peut être utilisé pour afficher des informations, agir sur les variables d'environnement, lancer des applications, connecter des lecteurs réseaux, lire et modifier la base de registre, ...

Ce projet a commencé en 1991 en réponse à de nombreuses demandes réclamant la mise à disposition d'un langage de script qui puisse interagir avec l'environnement réseau de Windows.

KiXtart a la particularité d'être sous licence *CareWare*. Cette licence est aussi appelée *CharityWare* ou encore *DonationWare*. Elle stipule que vous pouvez gratuitement télécharger KiXtart, l'installer et l'évaluer. Par contre, si vous l'utilisez régulièrement, vous êtes invités à faire un don à une association comme l'UNICEF.

2. Installer KiXtart

A présent, nous allons installer la dernière version à ce jour de KiXtart : KiXtart 2010 version 4.51.

- Télécharger sur KiXtart (647 Ko).

KiXtart est composé de 5 fichiers principaux :
- *Kix32.exe* (ou *Wkix32.exe*), le programme principal
- *Kx16.dll*, une DLL (Dynamic Link Library) 16 bits, utilisé pour se connecter au fichier *netapi.dll* sur les clients Windows 9x
- *Kx32.dll*, une DLL 32 bits, utilisé pour se connecter à netapi.dll sur les clients Windows 9x
- *Kxrpc.exe*, service RPC pour les clients Windows 9x
- *Kx95.dll*, une DLL 32 bits, utilisé par les clients Windows 9x pour se connecter au service RPC de KiXtart

Le programme *Kix32.exe* est une version console alors que *Wkix32.exe* est une version Windows. La version Windows affichera seulement la console s'il y a une sortie à l'écran.

Vous pouvez utiliser le paramètre *I* pour forcer la version Windows à ne pas afficher de console même s'il y a une sortie à l'écran.

Tous ces fichiers peuvent être installés et lancés à partir du disque dur en local ou par le réseau.
Pour installer KiXtart sur le réseau, vous devez copier certains fichiers dans le répertoire *NETLOGON* du serveur. Pour des clients Windows NT/2000/XP, vous devez copier le fichier *kix32.exe*. Pour des clients Windows 9x, copiez les fichiers *kx16.dll*, *kx32.dll* et *kix32.exe*. Si des clients Windows 9x utilisent les services RPC pour communiquer vous devez également copier le fichier *Kx95.dll*.

RPC, *Remote Procedure Call*, est un protocole utilisé dans le modèle client-serveur. Il gère les interactions entre le client et le serveur.

3. Mettre à jour et désinstaller KiXtart

Pour mettre à jour votre version de KiXtart, il faut simplement remplacer les fichiers que l'on a vu précédemment selon la version Windows des clients. Si le service RPC de KiXtart est utilisé, vous devez alors arrêter le service avec la commande :

```
net stop kxrpc
```

Puis, remplacer le fichier *kxrpc.exe* et redémarrer le service.

Pour désinstaller KiXtart, il suffit de supprimer les différents fichiers et scripts. Si vous utilisez le service RPC de KiXtart, vous devez taper la commande :

```
kxrpc -remove
```

4. Lancer un script avec KiXtart

KiXtart peut être lancé manuellement ou automatiquement à l'ouverture d'une session.

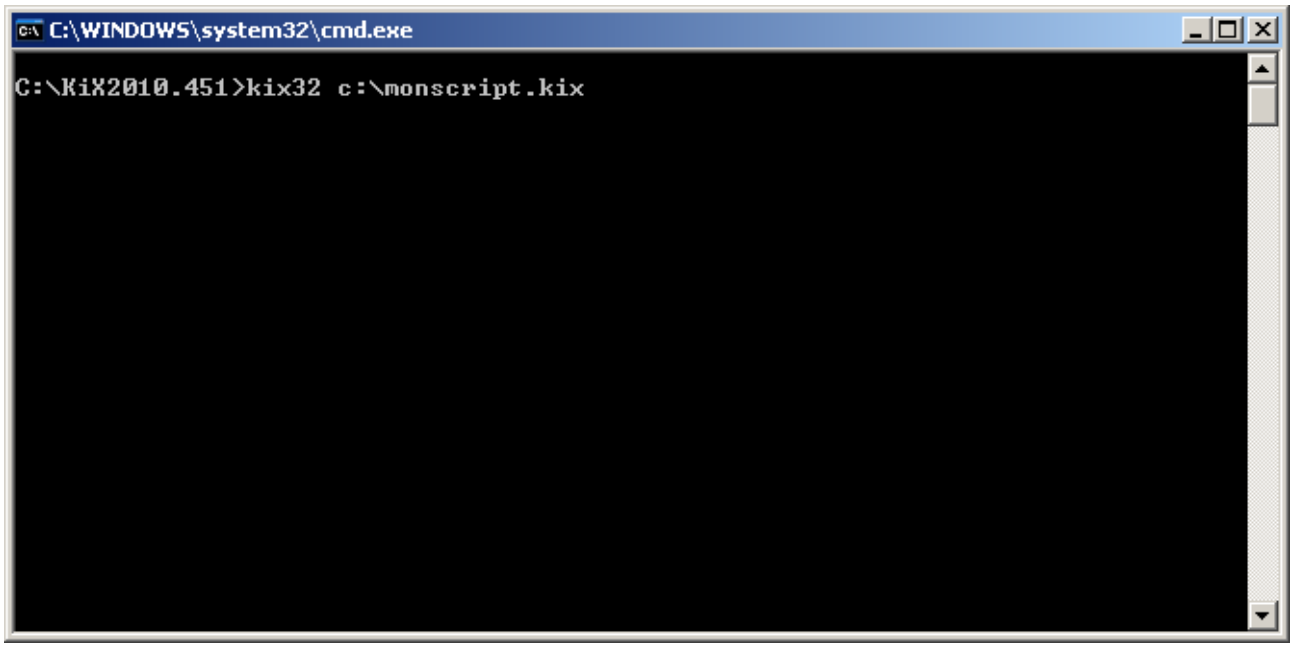
Pour lancer KiXtart manuellement, il suffit de lancer *kix32.exe* suivi de votre script. A noter qu'un script KiXtart est simplement un fichier texte ayant comme extension *KX* ou *KIX*.

Les anciennes versions de KiXtart pouvaient aussi utiliser l'extension *SCR*.

Par exemple, pour lancer le script *monscript.kix*, il suffit alors de taper dans la console :

```
kix32 c:\monscript.kix
```

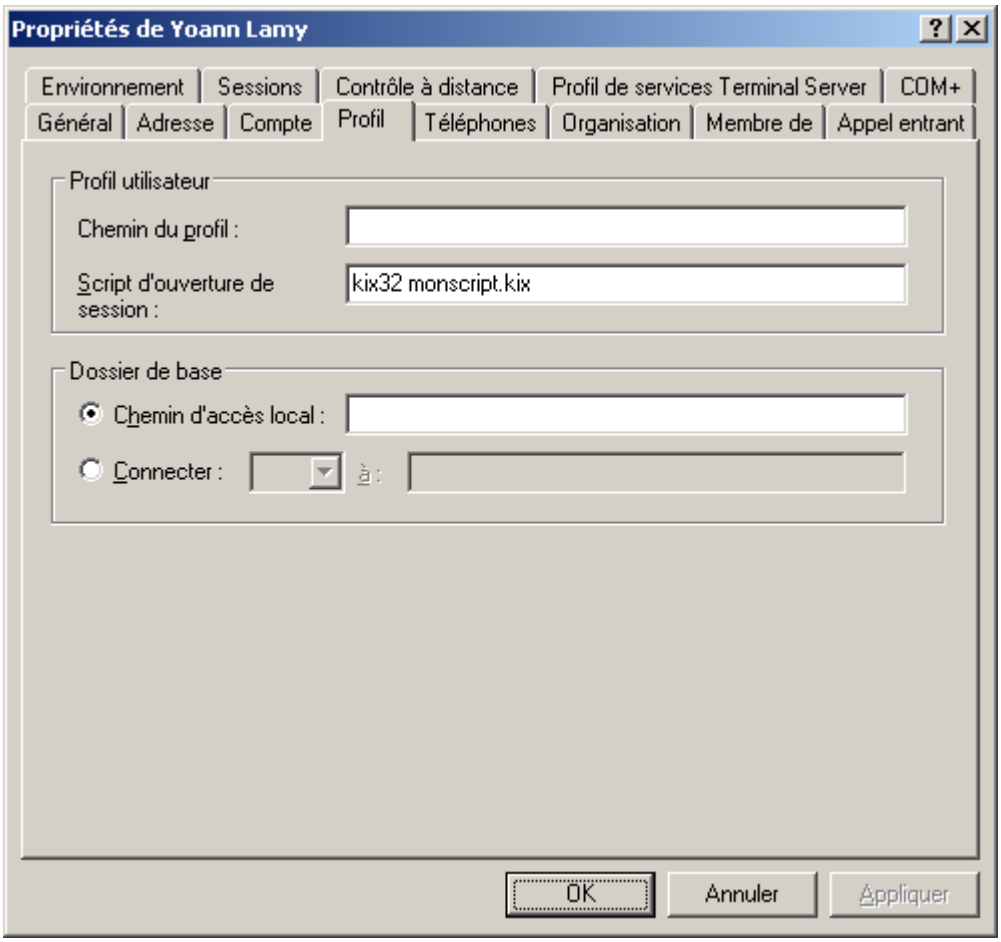
L'erreur fréquente est d'oublier de spécifier le chemin du script. KiXtart vous retournera alors une erreur : *ERROR : failed to find/open script*.



Le programme principal *kix32.exe* avec lequel nous lançons les scripts, possède quelques paramètres. Pour accéder à la description de ces paramètres, il suffit alors de taper :

```
kix32 -?
```

Pour lancer automatiquement un script KiXtart à l'ouverture d'une session, vous pouvez renseigner l'onglet *Profil* dans les propriétés des utilisateurs dans la gestion des utilisateurs de Windows.



KiXtart et le script *monscript.kix* se trouvent dans le répertoire partagé *NETLOGON*.

Si vous rencontrez des problèmes avec KiXtart, référez-vous à la documentation officielle (*kix2010.pdf*, page 16).

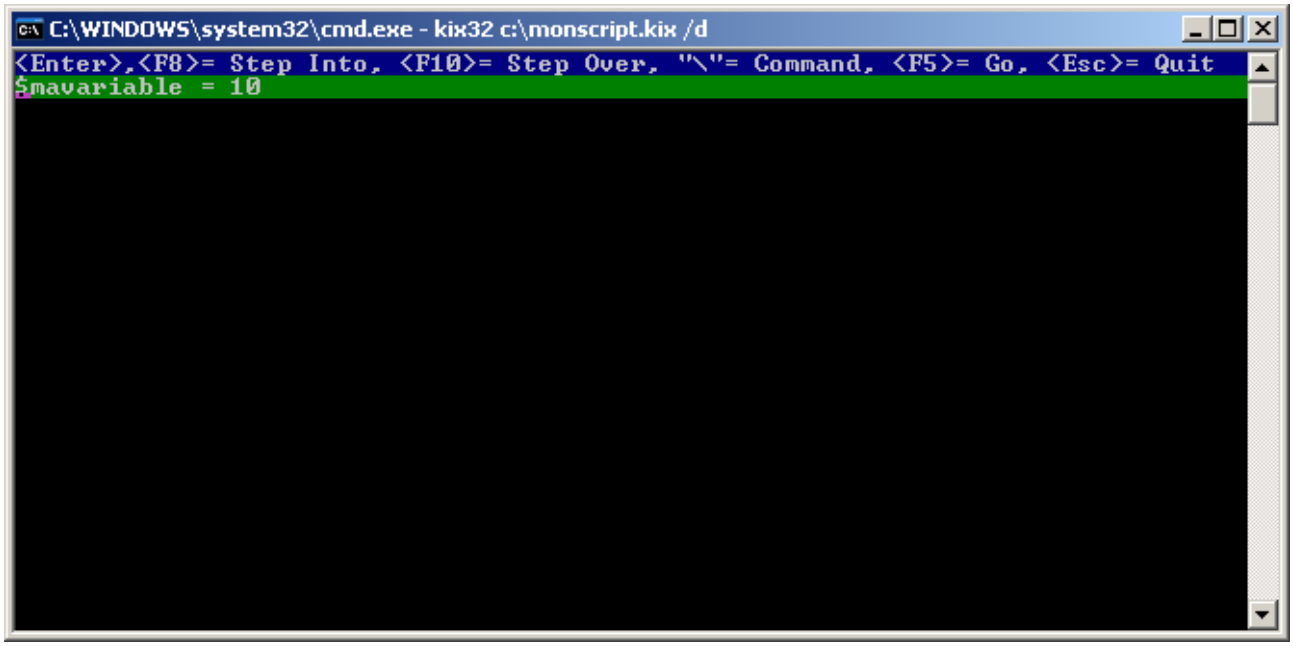
5. Lancer un script avec KiXtart en mode débogage

Ce mode exécute chaque instruction les unes après les autres pour trouver et corriger les erreurs de code. On appelle aussi cela le mode pas à pas. Pour lancer votre script dans ce mode, il suffit d'ajouter le paramètre *d* à KiXtart.

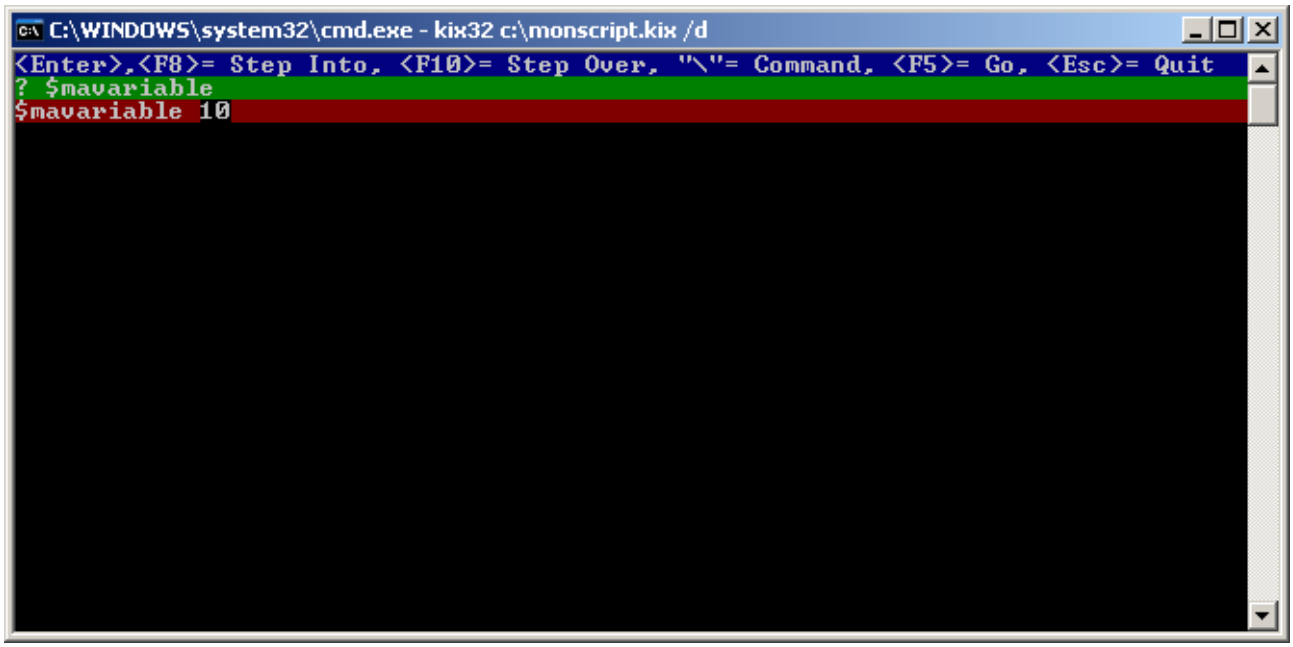
```
Kix32 c:\monscript.kix /d
```

Vous pouvez aussi lancer ce mode dans votre script avec la commande *Debug on*.

Le script est alors exécuté pas à pas. La ligne surlignée en vert affiche la ligne en cours d'exécution.



Les touches *F8* et *F10* servent à avancer ou reculer dans le mode pas à pas. La touche *F5* permet de terminer le mode pas à pas et continue d'exécuter normalement le reste du script. La touche *Esc* permet de quitter KiXtart.
La touche \ permet d'évaluer le contenu d'une variable, une macro ou un élément d'un tableau.



Spécifiez par exemple le nom de votre variable et appuyez sur la touche *Entrée* de votre clavier pour obtenir sa valeur.

6. Précompiler un script KiXtart

A partir de KiXtart version 4.5, vous pouvez précompiler un script. Cette option est pratique si vous souhaitez que votre script ne puisse pas être lu par n'importe quelle personne.

Pour réaliser cette opération, il vous suffit de lancer KiXtart en spécifiant le paramètre *t*.

```
Kix32 c:\monscript.kix /t
```

KiXtart générera alors un fichier d'extension *KX* : *c:\monscript.kx*. Vous lancerez ensuite ce script précompilé de la même manière qu'un simple script.

Une autre option permet de précompiler et aussi d'indiquer un mot de passe avec le paramètre *u*.

```
Kix32 c:\monscript.kix /t /u=MotDePasse
```

Par la suite, vous devrez donner un mot de passe pour pouvoir lancer votre script précompilé.

```
Kix32 c:\monscript.kx /u=MotDePasse
```

7. Le langage

KiXtart n'est pas sensible à la casse. Ce qui signifie qu'il ne fait pas de différence entre les minuscules et les majuscules.

7.1 Les commentaires

Pour commenter une ou plusieurs lignes, il suffit de placer les caractères *;* ou /* et */ .

```
; Ceci est un commentaire sur une ligne

 /* Ceci est un commentaire
sur plusieurs lignes*/
```

7.2 Les chaînes de caractères

Les chaînes de caractères peuvent contenir n'importe quels caractères excépté */0* (null) et */xla* (fin de fichier). Les chaînes de caractères doivent être séparées par des quotes (simple ou double).

```
? "ma chaîne de caractères"
```

Cet exemple affiche dans la console une nouvelle ligne contenant le texte entre double quotes.

7.3 Les opérateurs

KiXtart utilise les opérateurs de comparaison et logiques standard, à savoir : < (inférieur), > (supérieur), = (égal), <> (différent), <= (inférieur ou égal), >= (supérieur ou égal), *NOT* (NON

logique), *AND* (ET logique) et *OR* (OU logique).
On retrouve aussi les opérateurs numériques : +, -, *, /, *mod* (division pour récupérer le reste), *&* (ET binaire), | (OU binaire), ^ (OU exclusif binaire), ~ (NON binaire).

7.4 Les variables

On distingue deux types de variables :
- Les variables pré-définies, que l'on appelle aussi des macros.
- Les variables dynamiques.

Les variables d'environnement peuvent également être manipulées. Elles sont entourées par le caractère *%*.
Exemple : *? "Afficher le chemin complet de la console de commandes : %ComSpec%"*

7.4.1 Les macros

Les macros sont définies à l'aide du caractère @. Il en existe aujourd'hui une soixantaine. Elles permettent d'obtenir des informations sur le système comme l'heure, le nom de domaine ou l'adresse IP. Quelques exemples de macros : *@Date*, *@Domain*, *@HomeDir*, *@IPAdressX*, ...

```
? "La version de Windows est @ProductType @CSD"
? "Le nom de l'ordinateur est @HostName"
? "Le nom de l'utilisateur est @UserID"
? "Le nom de domaine ou groupe de travail est @Domain"
? "L'adresse IP est @IPaddress0"
? "L'adresse IP de la passerelle par défaut est @IPaddress1"
```

Vous pouvez obtenir la liste complète des macros à cette adresse ou en consultant l'aide de KiXtart.

Si vous souhaitez utiliser le caractère @ dans une chaîne par exemple pour indiquer une adresse mail, vous devrez alors en mettre deux successivement
Exemple : *? "monnom@@monadresse.fr"*

7.4.2 Les variables dynamiques

Les variables dynamiques permettent de stocker temporairement des valeurs durant l'exécution d'un script. Elles sont définies à l'aide du caractère *$*. Le nom des variables que vous attribuez ne doit pas contenir d'opérateurs logiques (+, -, *, /, &, <, >, =).

```
$mavariable = 10
```

Les variables dynamiques peuvent aussi être assignées lors du lancement du script.

```
kix32 c:\monscript.kix $mavariable=10
```

L'erreur fréquente est l'insertion d'espaces entre le signe d'égalité. Si vous voulez mettre des espaces, vous devez alors utiliser les doubles quotes.
Exemple : *kix32 c:\monscript.kix $mavariable="Bonjour, ça va"*

Les variables dynamiques peuvent être déclarées de deux façons : DIM et GLOBAL.

Pour les variables déclarées avec GLOBAL, elles seront accessibles dans tout le script. Par contre, les variables déclarées avec DIM, appelées aussi variables locales, pourront être manipulées seulement dans une partie de code. Par exemple, si vous déclarez une variable de type DIM dans une fonction, celle-ci ne pourra pas être accessible à l'extérieur de cette fonction.

```
DIM $mavariable=10
```

En outre, par défaut, les variables dynamiques sont implicites. Ce qui signifie qu'il n'est pas nécessaire de les déclarer. Elles seront alors déclarées automatiquement lors d'une assignation.

Il est possible de demander à KiXtart que toutes les variables soient explicites (obligation de les déclarer) avec *SetOption*. Pour plus d'informations sur cette commande, veuillez consulter l'aide de KiXtart.

Dans KiXtart, les variables sont de type variant. Ce type de données permet de s'adapter à la valeur de tout type sans avoir recours à des conversions. En réalité KiXtart utilise trois types de données : les chaînes de caractères, les entiers relatifs (signés) et les nombres à virgule flottante. Une chaîne de caractères peut contenir 32 000 caractères, un entier relatif peut contenir une valeur comprise entre -2 147 483 648 et 2 147 483 647 et un nombre à virgule flottante d'une longueur de 8 octets (64 bits).

A noter que KiXtart peut gérer d'autres types de données qui peuvent être employées par d'autres programmes.

7.5 Les tableaux

Un tableau permet de contenir une série de variables. KiXtart supporte les tableaux jusqu'à 60 dimensions.

```
DIM $montableau[14]
```

Le tableau ainsi créé, est appelé un tableau à une dimension. Il sera composé de 15 éléments (de 0 à 14).
On peut aussi déclarer implicitement les tableaux.

```
$montableau1 = 10, 20, 30
$montableau2 = "A1", "B2", "C3"
```

Il y aura alors trois éléments dans ces deux tableaux (de 0 à 2).

$montableau1

10	20	30
0	1	2

$montableau2

A1	B2	C3
0	1	2

Ecrit explicitement le code de ces deux tableaux donnerait :

```
DIM $montableau1[2]
$montableau2[2]

$montableau1[0] = 10
$montableau1[1] = 20
$montableau1[2] = 30

$montableau2[0] = "A1"
$montableau2[1] = "B2"
$montableau2[2] = "C3"
```

Pour accéder aux valeurs de ces deux tableaux, vous devez simplement indiquer leur identifiant.

```
$montableau1[0] ;La valeur sera 10
$montableau1[1] ;La valeur sera 20
$montableau1[2] ;La valeur sera 30

$montableau2[0] ;La valeur sera A1
$montableau2[1] ;La valeur sera B2
$montableau[2] ;La valeur sera C3
```

Si vous voulez accéder à une valeur d'un tableau qui n'existe pas, KiXtart vous retournera alors une erreur : *array reference out of bounds!*

On peut également créer des tableaux à deux dimensions comme ceci :

```
DIM $montableau[20,3] ;Tableau de 21 lignes (de 0 à 20 ) et 4 colonnes (de 0 à 3)
```

Ce tableau compte donc 84 éléments.

7.6 Les conditions

Les conditions permettent d'effectuer des tests. On utilise la structure *if*, *else* et *endif*.

```
if  expression
...
else
...
endif
```

Voici un petit exemple de condition.

```
$i = 5 ;Exemple de valeur
if $i = 5 ;Si la variable $i est égale à 5 ...
  ? "La valeur est égale à 5" ;On affiche un message
else ;Sinon ...
  ? "La valeur n'est pas égale à 5" ;On affiche un autre message
endif
```

Il existe aussi le mot clé *iif* qui permet de retourner une des deux valeurs suivant le résultat de la condition.

```
$i = 5 ;Exemple de valeur
iif ($i =5, "La valeur est égale à 5", "La valeur n'est pas égale à 5")
```

Les deux codes sont équivalents.

Si vous souhaitez effectuer plusieurs conditions pour le même traitement, au lieu d'utiliser plusieurs fois le couple *if* et *endif* à la suite, vous pouvez utiliser *Select*, *Case* et *EndSelect*.

```
$i = 10 ;Exemple de valeur

select
  case $i = 5 ;Si la variable $i est égale à 5 ...
    "La valeur est égale à 5" ;On affiche un message
  case $i < 5 ;Si la variable $i est inférieure à 5 ...
    "La valeur est inférieure à 5" ;On affiche un autre message
  case 1 ;Sinon ...
    "La valeur est supérieure à 5" ;On affiche encore un autre message
endselect
```

7.7 Les boucles

Les boucles permettent de répéter un bloc de codes un certain nombre de fois. KiXtart propose quatre types de boucles.

7.7.1 La boucle Do Until

La première que nous verrons est la boucle *Do Until*. Cette boucle contient une expression conditionnelle de contrôle qui se poursuit jusqu'à ce que la condition devienne vraie.

```
do
...
until expression
```

Cet exemple montre le principe de cette boucle.

```
$i = 1
do
  ? "La variable vaut $i"
  $i = $i + 1
until $i = 5
? "La boucle est terminée car la variable vaut $i"
```

7.7.2 La boucle While Loop

La deuxième boucle *While Loop* exprime aussi une condition de continuation. Cette boucle s'arrête lorsque la condition devient fausse (l'inverse de la boucle *do until*).

```
while expression
...
loop
```

L'exemple suivant effectue la même opération que précédemment.

```
$i = 1
while $i <> 5
  ? "La variable vaut $i"
  $i = $i + 1
loop
? "La boucle est terminée car la variable vaut $i"
```

7.7.3 La boucle For Next

La boucle *For Next* connaît à l'avance le nombre de répétitions grâce a une variable. Vous pouvez aussi spécifier par *step* le pas d'incrémentation ou de décrémentation (indiquer alors un signe moins pour le pas).

```
for ... to ... step ...
...
next
```

Un exemple de cette boucle pour comprendre le fonctionnement.

```
for $i = 1 to 5 step 1
  ? "La variable vaut $i"
next
? "La boucle est terminée car la variable vaut $i"
```

Si le pas vaut 1, il n'est alors pas nécessaire de le renseigner.

```
for $i = 1 to 5
  ? "La variable vaut $i"
next
? "La boucle est terminée car la variable vaut $i"
```

7.7.4 La boucle For Each

La boucle *For Each* permet de répéter une portion de code pour chaque élément d'un tableau.

```
for each ... in ...
...
next
```

Par exemple, on peut afficher le contenu d'un tableau.

```
DIM $montableau[2]
$montableau[0] = 10
$montableau[1] = 20
$montableau[2] = 30
for each $element in $montableau
  ? $element
next
```

7.8 Création d'une fonction

Une fonction est un ensemble d'instructions qui est susceptible d'être appelé à plusieurs reprises dans un script. Une fonction est constituée d'un nom et d'arguments. Elle renvoie habituellement une valeur.

```
function MaFonction($MaChaine)
? $MaChaine
endfunction

MaFonction("Bonjour") ;Appel la fonction "MaFonction" avec "Bonjour" en argument
```

Vous pouvez bien entendu ajouter plusieurs arguments à une fonction simplement en les séparant par des virgules. Pour renvoyer une valeur, il suffit d'utiliser le nom de la fonction comme variable.

```
function MaFonction($Nombre1, $Nombre2)
  $MaFonction = $Nombre1 + $Nombre2
endfunction

? MaFonction(2, 3)
```

7.9 Sous-programme

Il est possible de faire appel à un sous-programme dans son script à l'aide de la commande *GoSub* suivit d'un label. On retrouvera dans la suite ce même label avec son code suivi du mot clé *Return* permettant de revenir au programme principal.

```
? "Cet exemple appel un sous-programme"
GoSub "test"
? "Fin de l'exemple"
Exit 1

:test
  ? "Nous sommes dans le sous-programme"
Return
```

Il est nécessaire ici d'indiquer avec *Exit 1* la fin du programme pour que le sous-programme ne s'exécute pas deux fois.

Il existe également la commande *Goto* permettant d'aller vers un sous-programme.

```
? "Cet exemple appel un sous-programme"
Goto "test"
? "Fin de l'exemple" ;Cette ligne ne sera pas affichée contrairement à GoSub

:test
  ? "Nous sommes dans le sous-programme"
```

La différence avec *GoSub* est que l'on ne retournera pas dans le programme principal.

7.10 Appeler un autre script KiXtart

A partir d'un script, on peut également appeler un second script grâce à la commande *Call* suivit du nom du fichier.

```
? "Début du script"
Call "c:\monscript2.kix" ;Appel un autre script
? "Fin du script"
```

Le deuxième script *monscript2.kix* se lancera dès que l'on fera appel à lui.

```
? "Je suis dans mon deuxième script"
```

A la fin du second script ou si l'on rencontre le mot clé *Return*, on reviendra alors dans le premier script.

A partir de KiXtart 2010 version 4.51, il existe une nouvelle commande : *Include*. Son rôle est d'utiliser comme le fait la commande *Call* un autre script. Toutefois, il subsite quelques différences entre les deux. La grande différence est que la commande *Include* ne fait pas appel à un script mais place directement le script en question dans le script qui l'a appelé. Une variable locale déclarée dans un script sera donc accessible dans un autre script avec *Include* mais ne le sera pas avec un *Call*.

8. Lancer une application

Pour lancer une application à partir d'un script KiXtart, il existe deux commandes : *Run* et *Shell*.

Ces deux commandes permettent de lancer une application ou un interpréteur de commandes. La différence entre les deux est qu'avec la commande *Shell*, KiXtart attend que l'application ait fini de se lancer pour continuer le script.

```
Shell "C:\WINDOWS\notepad.exe"
```

Cet exemple lance l'éditeur de texte de Windows.

9. Les commandes systèmes

KiXtart intègre les commandes systèmes standard comme *CD*, *CLS*, *Copy*, *Del*, *MD*, *Move*, *RD*.

```
Copy "c:\test.txt" "c:\test2.txt"
```

Cet exemple effectue une simple copie de fichier.

10. Les entrées clavier

Un script a souvent besoin de dialoguer avec un utilisateur pour choisir par exemple entre deux options. Il existe pour cela les commandes *Get* et *Gets*.

La première commande permet de demander à l'utilisateur de n'entrer qu'un seul caractère.

```
? "Entrez un seul caractère : "
Get $c
? "Vous avez entré : $c"
```

La commande *Gets* permet d'entrer une chaîne de caractères. Pour que celle-ci soit validée, l'utilisateur doit appuyer sur la touche *Entrée* du clavier.

```
? "Entrer une chaîne de caractères : "
Gets $chaine
? "Vous avez entré : $chaine"
```

11. Les boîtes de messages

Une boîte de message est utilisée pour dialoguer de manière visuelle avec l'utilisateur.

```
MessageBox("Mon message", "titre", 32)
```

Le premier paramètre de cette commande permet d'indiquer le texte, le deuxième, le titre et le dernier paramètre est un nombre entier permettant de modifier l'apparence de la boîte.

Paramètre	***Valeur***
Icône d'erreur	16
Icône de confirmation	32
Icône d'avertissement	48
Icône d'information	64
Bouton OK	0
Boutons OK et Annuler	1
Boutons Abandonner, Réessayer et Ignorer	2
Boutons Oui, Non et Annuler	3
Bouton Oui et Non	4
Boutons Réessayer et Annuler	5
Choix du premier bouton par défaut	0
Choix du deuxième bouton par défaut	256
Choix du troisième bouton par défaut	512
La fenêtre se place en avant-plan	4096

Pour le troisième paramètre, vous devez additionner ces valeurs pour obtenir ce que vous désirez.

Pour insérer, plusieurs lignes dans un message, vous devez ajouter un retour à la ligne. Vous devez alors inclure : *Chr(13) + Chr(10)* ou simplement la macro *@CRLF*.

La réponse choisie par l'utilisateur renvoie alors en retour une valeur.

Choix de l'utilisateur	***Valeur***
Bouton OK	1
Bouton Annuler	2
Bouton Abandonner	3
Bouton Réessayer	4
Bouton Ignorer	5
Bouton Oui	6
Bouton Non	7

Le code suivant permet de tester le choix de l'utilisateur.

```
$choix = MessageBox("Mon message", "titre", 33)
if $choix = 1
  MessageBox("Bouton OK", "Choix", 64)
else
  MessageBox("Bouton Annuler", "Choix", 64)
endif
```

Il n'existe pas pour le moment de boîte de saisie comme sous Visual Basic Script avec la commande *InputBox*. Cependant, plusieurs techniques existent pour pallier à ce manque. La première est l'utilisation de KiXforms et la deuxième est l'utilisation d'applications extérieures comme Excel via l'automatisation COM.

12. Manipuler des chaînes de caractères

KiXtart dispose de commandes permettant de manipuler des chaînes de caractères.

On peut mettre un texte tout en majuscule ou en minuscule.

```
? UCase("Mon Texte En Majuscule")
? LCase("Mon Texte En Minuscule")
```

On peut récupèrer un mot particulier dans une chaîne.

```
$MonMot =  SubStr("Un texte d'exemple", 4, 5) ;Récupère le mot "texte" de la chaîne de caractères
? "Le mot '$MonMot' fait " + Len($MonMot) + " caractères" ;Compte le nombre de caractère du mot "texte"
```

Le caractère + permet de concaténer un ensemble de chaîne de caractères.

On peut aussi récupèrer les *X* premiers ou derniers caractères d'une chaîne.

```
? Left("Ceci est un texte d'exemple", 4) ;Récupère les 4 premiers caractères
? Right("Ceci est un texte d'exemple", 7) ;Récupère les 4 derniers caractères
```

On peut rechercher un mot particulier dans une chaîne.

```
if InStr ("Ceci est un texte d'exemple", "texte") = 0
  ? "Le mot 'texte' n'a pas été trouvé"
else
  ? "Le mot 'texte' a été trouvé"
endif
```

13. Les fichiers

Les scripts KiXtart peuvent travailler avec des fichiers de tout type. Plusieurs commandes existent pour permettre d'accèder à des informations sur des fichiers.

On peut par exemple récupèrer la taille d'un fichier, récupèrer la date et l'heure de la dernière modification du fichier.

```
$file = "C:\test.txt" ;Chemin du fichier
if Exist($file) ;Si le fichier existe ...
  $espace = GetFileSize($file) ;Récupère la taille d'un fichier (en Octet)
  $espace = CDbl($espace) / 1024 ;Transforme la taille en Ko
  $espace = FormatNumber ($espace, 2) ;Deux chiffres après la virgule
  ? "La taille du fichier est de $espace Ko" ;Affichage de la taille du fichier
  ? "La date de modification du fichier est " + GetFileTime($file) ;Récupère la date et l'heure de modification du fichier (date au format anglais)
else
 ? "Le fichier est introuvable" ;Affiche un message pour indiquer que le fichier n'existe pas
endif
```

Comme nous l'avons vu précédemment, les variables sont de type variant. KiXtart se base sur la première variable (dans ce cas, la variable *$espace*) pour définir quel sera le type pour le résultat de l'affectation. Nous sommes donc obligés d'utiliser la commande *CDbl* si l'on souhaite obtenir le résultat sous forme d'un nombre à virgule.

On peut aussi récupérer les informations de versions d'un fichier exécutable avec la commande *GetFileVersion*.

```
$file = "C:\KiX2010.451\Kix32.exe" ;Chemin du fichier
if Exist($file) ;Si le fichier existe ...
  ? "Description du fichier : " + GetFileVersion($file, "FileDescription")
  ? "Version du fichier : " + GetFileVersion($file, "FileVersion")
  ? "Nom de l'entreprise : " + GetFileVersion($file, "CompanyName")
else
 ? "Le fichier est introuvable" ;Affiche un message pour indiquer que le fichier n'existe pas
endif
```

La commande *SetFileAttr* sert à placer des attributs à un fichier.

```
$file = "C:\test.txt" ;Chemin du fichier
if SetFileAttr($file, 1) = 0 ;Met le fichier en lecture seule, si la fonction réussit...
  ? "Le fichier $file est maintenant en lecture seule" ;Affiche un message
```

14. Les fichiers texte

KiXtart dispose de commandes pour lire et écrire dans un fichier texte.
Cet exemple permet de lire le contenu d'un fichier texte.

```
if Open(3, "C:\test.txt") = 0 ;Ouvre le fichier texte
  $line = ReadLine(3) ;Lecture de la première ligne
  while @error = 0 ;Tant que l'on n'est pas à la fin du fichier...
    ? $line ;Affichage des lignes
    $line = ReadLine(3) ;Lecture des ligne suivantes
  loop
  Close(3)
else
  MessageBox("Le fichier n'a pas pu être créé", "Avertissement", 48) ;Affiche un message
d'avertissement
endif
```

Le chiffre *3* que l'on rencontre dans ce code correspond à un numéro d'identification que vous attribuez au fichier texte.

@error est une macro particulière. Elle permet de savoir si une erreur est intervenue lors de la dernière commande. La valeur *0* signifie que la commande a réussi. N'importe quelle autre valeur indique une erreur.

Pour afficher le contenu d'un fichier dans la console, on peut aussi utiliser la commande *Display*.

Il est souvent nécessaire de créer et d'écrire dans un fichier texte pour conserver une trace des actions effectuées.

```
if Open(3, "C:\log.txt", 5) = 0 ;Crée le fichier s'il n'existe pas et le prépare en écriture
  WriteLine(3, "Le fichier a été créé le @date  à @time @crlf") ;Ecrit une ligne dans le fichier
  Close(3) ;Fermeture du fichier
else
  MessageBox("Le fichier n'a pas put être créé", "Avertissement", 48) ;Affiche un message
d'avertissement
endif
```

@crlf correspond aux caractères de fin de ligne.

15. La base de registre

Il est possible en effet avec KiXtart d'agir sur la base de registre. On peut par exemple cacher le

lecteur de disquette en allant écrire une valeur dans la base de registre.

```
$Key =
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer"
if WriteValue($Key, "NoDrives", "1", "REG_DWORD") = 0 ;Ajoute une nouvelle valeur
  ? "Le lecteur de disquette sera maintenant caché" ;Affiche un message si l'opération est réussie
endif
```

Certaines modifications de la base de registre nécessitent un redémarrage pour que celles-ci soient prises en compte.

KiXtart prend en charge les fichiers INI avec les commandes *ReadProfileString* et *WriteProfileString*.

16. Monter des lecteurs réseaux

Pour monter des lecteurs réseaux, on utilise non pas la commande *net use* (en batch) mais simplement *Use*.

```
Use Z: "\\Serveur\Repertoire"
```

On peut aussi monter un lecteur réseau suivant le groupe de l'utilisateur.

```
if InGroup("MonGroupe1")
  Use Z: "\\Serveur\Repertoire"
endif
```

On peut indiquer plusieurs groupes en les séparant par des virgules.

```
if InGroup("MonGroupe1", "MonGroupe2")
  Use Z: "\\Serveur\Repertoire"
endif
```

Si on souhaite par exemple que seul les utilisateurs membres des deux groupes puissent obtenir ce lecteur réseau, on indique alors un *1* dans *InGroup*.

```
if InGroup("MonGroupe1", "MonGroupe2", 1)
  Use Z: "\\Serveur\Repertoire"
endif
```

KiXtart possède un système de cache pour les groupes (à partir de Windows NT et des versions supérieures). Cela signifie qu'il garde en mémoire tous les utilisateurs qui se sont connectés et leurs groupes d'appartenance. Il actualise ceci automatiquement tous les 30 jours.

Si vous souhaitez forcer cette actualisation, vous devez alors indiquer le paramètre *f* lorsque vous lancez votre script avec KiXtart.

```
kix32 c:\monscript.kix /f
```

Vous pouvez aussi spécifier une date pour forcer cette actualisation.

```
kix32 c:\monscript.kix /f:2006/12/31
```

17. Synchronisation de l'horloge Windows

Pour synchroniser l'horloge Windows d'un ordinateur avec le réseau, on utilise la commande *SetTime* suivi soit du nom du serveur soit du nom du domaine ou soit du caractère *. Ce caractère indique que l'on recherche un serveur dans le domaine pour se synchroniser.

```
SetTime "\\NomDuServeur"
```

18. Automatisation COM

L'automatisation COM, *Component Object Model*, est une normalisation de développement objet, créé par Microsoft. Le but est de réutiliser certaines fonctionnalités d'autres applications comme Internet Explorer, Word, Excel, Outlook, ...

Cet exemple lance une page Web avec Internet Explorer.

```
$IE = CreateObject("InternetExplorer.Application") ;Crée un objet
if @error = 0 ;S'il n'y a pas d'erreur ...
  $IE.Visible = 1 ;Affiche l'application
  $IE.Navigate("http://ww.google.fr/") ;Lance une page Web avec Internet Explorer
else ;S'il y a une erreur ...
  ? "Une erreur est survenue" ;On affiche un message
endif
```

19. Dessiner dans la console

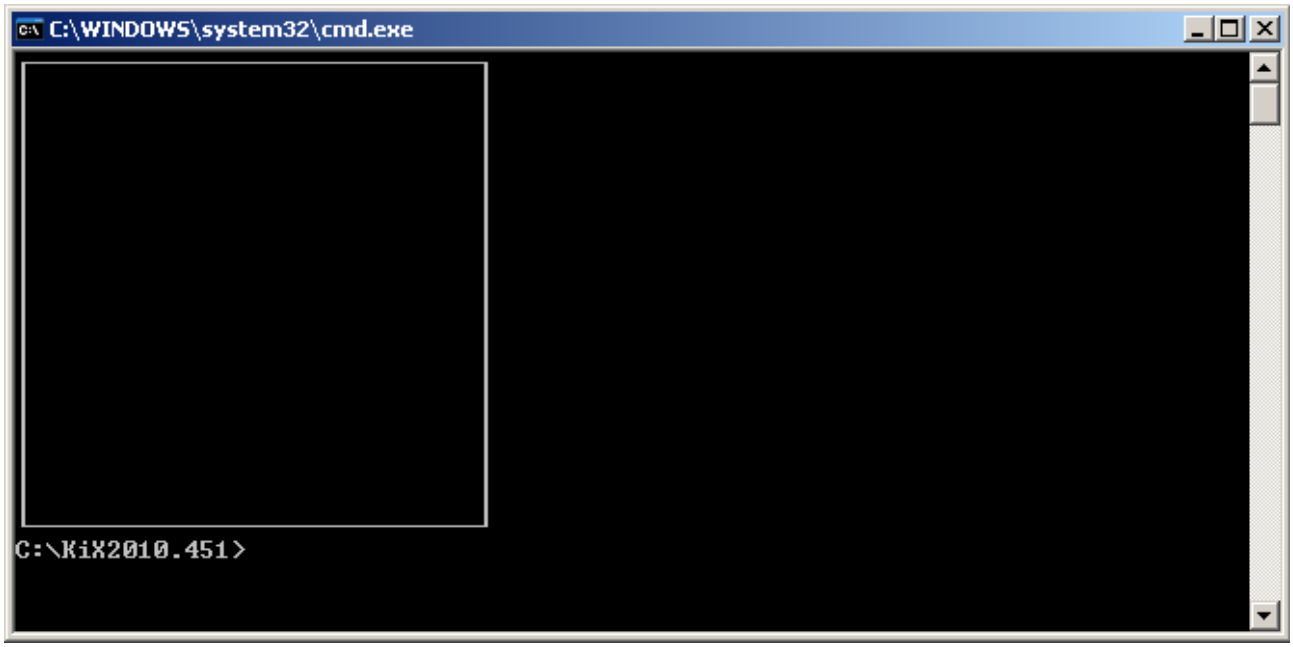
Il est en effet possible de dessiner dans la console. Pour cela, on utilise la commande *Box*.

```
Box (top_left_row, top_left_column, bottom_right_row, bottom_right_column, "line style")
```

Les quatre premiers paramètres expriment la dimension et le positionnement de la boîte. Le dernier paramètre indique le type de ligne de cette boîte.

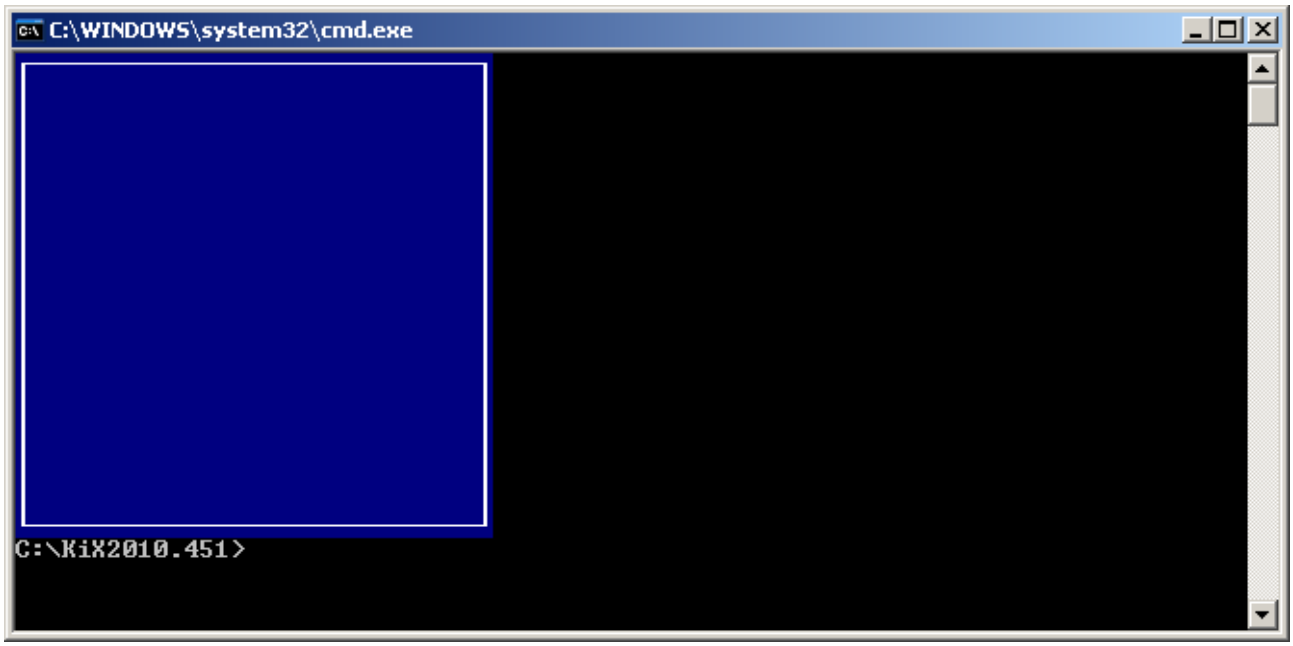
Style de ligne	***Valeur***
Ligne simple	single
Ligne double	double
Ligne épaisse	full
Grille	grid

```
Box(0, 0, 20, 30, "single")
```



On peut aussi ajouter de la couleur à l'aide de la commande *color*.

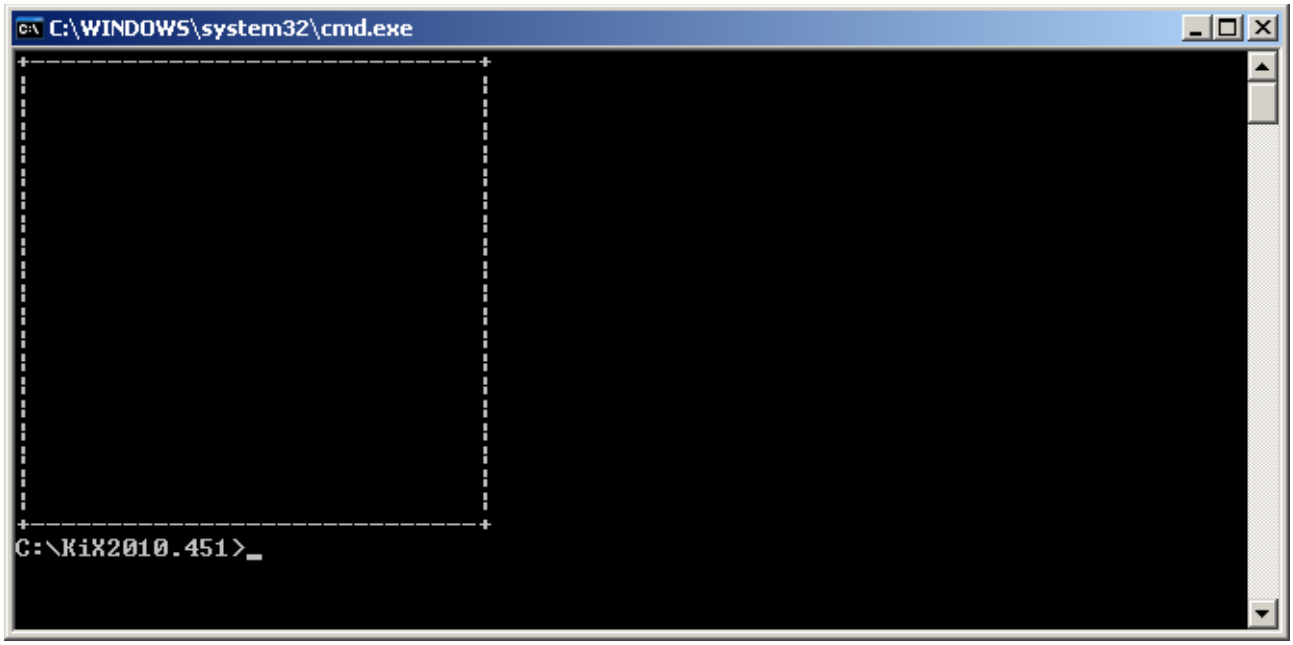
```
Color w+/b ;Texte en blanc sur fond bleu
Box(0, 0, 20, 30, "single")
```



Pour plus d'information sur cette commande, vous pouvez consulter l'aide.

Il est possible aussi de personnaliser le style de ligne avec des caractères.

```
Box(0, 0, 20, 30, "+-+|+-+| ")
```



On peut très bien ajouter du texte.

```
Cls ;Efface la console
Box(0, 0, 10, 73, "+-+|+-+| ")
Color b+/n ;Texte de couleur bleu
Big ;Agrandit les caractères
At (2,10) "KiXtart" ;Place le curseur dans la position indiquée (2 de hauteur et 10 à gauche)
At (11,0) ;
```

Ces commandes, vous l'aurez compris restent tout de même assez limitées.

20. KiXforms

KiXforms est une extension de KiXtart permettant de créer des contrôles comme des fenêtres ou boutons.

KiXforms fonctionne sous Windows 9x/NT/2000/XP et 2003. Cependant, il se peut qu'il y est quelques bugs mineurs sous Windows 9x.

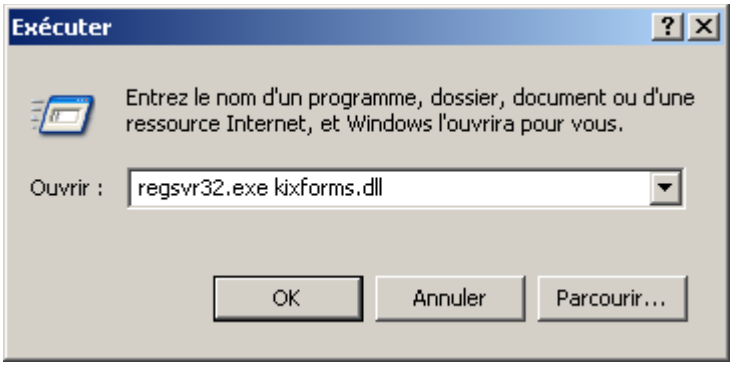
20.1 Installation de KiXforms

Télécharger la dernière version (à ce jour, v2.30) de KiXforms.

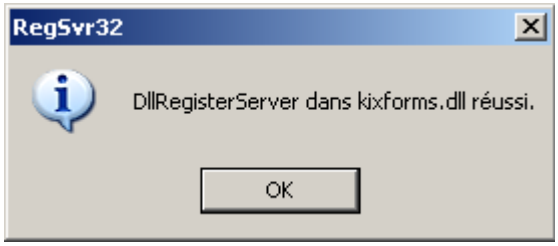
Pour installer KiXforms, vous devez placer sur chaque client le fichier *kixforms.dll* dans un répertoire système (par exemple, c:\WINDOWS\System\).

Nous allons maintenant enregistrer cette DLL. Allez dans le menu *Démarrer* de Windows puis sur *Exécuter*. Tapez cette ligne :

```
regsvr32.exe kixforms.dll
```



Cliquez sur le bouton *OK*. Une fenêtre vous informe alors que la librairie a bien été enregistrée.



Si vous souhaitez mettre à jour ou supprimer KiXforms, vous devez tout d'abord retirer la DLL, en tapant :

```
regsvr32.exe /u kixforms.dll
```

20.2 Créer une fenêtre avec KiXforms

```
Break On
$System = CreateObject("Kixtart.System")
$Form = $System.Form()
$Form.Text = "Mon Titre" ;Titre de la fenêtre
$Form.Width = 320 ;Largeur de la fenêtre
$Form.Height = 200 ;Hauteur de la fenêtre
$Form.Center ;Centre la fenêtre par rapport à l'écran
$Form.BackColor = 0,0,255 ;Couleur de la fenêtre (bleu)
$Form.Show ;Affiche la fenêtre
While $Form.Visible ;Affiche la fenêtre tant que l'on n'a pas fermé
  $ = Execute($Form.DoEvents())
Loop

Exit 1
```

Si vous désirez que la console n'apparaisse pas en arrière-plan, il faut alors lancer le script avec le programme *Wkix32.exe* en spécifiant *I* en paramètre.

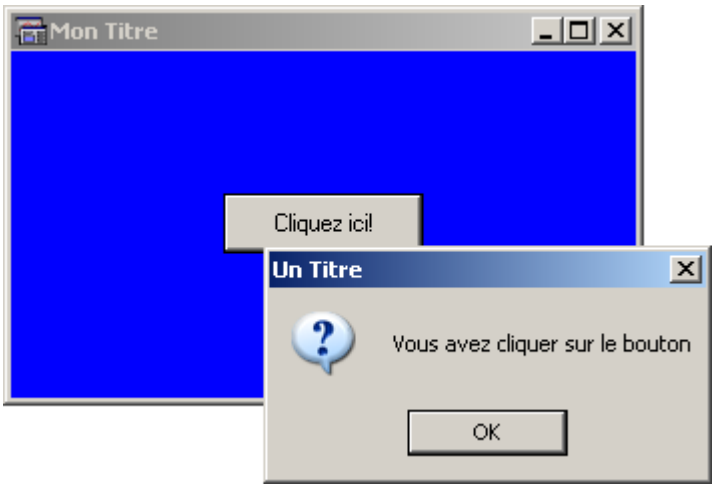
```
Wkix32 c:\monscript.kix /i
```

20.3 Insérer un bouton dans la fenêtre

```
Break On
$System = CreateObject("Kixtart.System")
$Form = $System.Form()
$Form.Text = "Mon Titre" ;Titre de la fenêtre
$Form.Width = 320 ;Largeur de la fenêtre
$Form.Height = 200 ;Hauteur de la fenêtre
$Form.Center ;Centre la fenêtre par rapport à l'écran
$Form.BackColor = 0,0,255 ;Couleur de la fenêtre (bleu)
$Button1 = $Form.Controls.Button() ;Création du bouton
$Button1.Text= "Cliquez ici!"
$Button1.Center ;Centre le bouton par rapport à la fenêtre
$Button1.OnClick = "Button1_Click()" ;Associe l'événement du bouton à la fonction
'Button1_Click()'
$Form.Show ;Affiche la fenêtre
While $Form.Visible ;Affiche la fenêtre tant que l'on n'a pas fermé
  $ = Execute($Form.DoEvents())
Loop

Exit 1

function Button1_Click()
  MessageBox("Vous avez cliqué sur le bouton", "Un Titre", 32)
endfunction
```



Certaines commandes sont dites *deprecated*. Ce qui signifie qu'elles sont devenues obsolètes. Elles ne seront peut être plus prises en compte dans les nouvelles versions de KiXforms. Attention donc aux scripts que vous pouvez trouver sur Internet.

20.4 KiXforms Designer

KiXforms Designer est une application permettant via une interface graphique de placer des contrôles sur une fenêtre et de générer le code correspondant.

21. Quelques liens

Pour terminer cette documentation, voici une liste de liens qui vous seront utiles pour l'élaboration de vos prochains scripts.

Lien	*Description*
http://www.kixtart.org/	Site officiel de KiXtart.
http://www.kixforms.org/	Site officiel de KiXforms. Cette librairie permet de créer des contrôles Windows comme des fenêtres, boutons, ...
http://www.gardinertrane.com/kixtarter/	Kixtarter est un éditeur de scripts avec coloration syntaxique.
http://www.kixscripts.com/	Nombreux scripts en tout genre.
http://www.kixtart.org/UDF/	Collection de scripts KiXtart.
http://www.microsoft.com/technet/scriptcenter/scripts/kixtart/default.mspx	Annuaire de scripts KiXtart.