Création, enregistrement et lecture d'un fichier de données à accès aléatoire

Le fichier à accès *random* (à la demande), aussi nommé fichier à accès direct, ou relatif, permet l'enregistrement de structures complexes. Les champs de chaque enregistrement, et donc les enregistrements eux-mêmes, doivent être tous de la même longueur. Un tel fichier peut être lu séquentiellement, mais aussi par accès direct à un enregistrement souhaité, notamment pour permettre le remplacement d'un enregistrement modifié. C'est la classe **`FileStream`** qui fournit les outils nécessaires.

Le fichier est une instance de la classe **`FileStream`** et son ouverture est réalisée par le constructeur selon les mode et type d'accès qui lui sont passés par paramètres.

```
Dim Fichier As FileStream
Dim NomFichier As String = "X:\MesDonnees\MonFichier.dat"
Fichier = New FileStream(NomFichier, FileMode, FileAccess)
```

Les modes d'ouvertures

Le mode d'ouverture d'un fichier est donné par une des valeurs possibles du paramètre **`FileMode`** du constructeur.

`Append`	Ouverture d'un fichier en ajout et positionnement pour écriture à la fin. Si le fichier n'existe pas, il est créé. Le mode d'accès **`FileAccess.Write`** est le seul permis sous ce mode d'ouverture.
`Create`	Ouverture d'un fichier en création et positionnement pour écriture au début. Si le fichier existe, il est remplacé. Le mode d'accès **`FileAccess.Read`** est interdit sous ce mode d'ouverture.
`CreateNew`	Ouverture d'un fichier en création et positionnement pour écriture au début. Si le fichier existe, une exception est générée. Le mode d'accès **`FileAccess.Read`** est interdit sous ce mode d'ouverture.
`Open`	Ouverture d'un fichier existant. Si le fichier n'existe pas, une exception est générée.
`OpenOrCreate`	Ouverture d'un fichier existant ou création s'il n'existe pas.
`Truncate`	Ouverture et vidange d'un fichier existant. Si le fichier n'existe pas, une exception est générée. Le mode d'accès **`FileAccess.Read`** est interdit sous ce mode d'ouverture.

Les types d'accès

Le type d'accès à un fichier est donné par une des valeurs possibles du paramètre **`FileAccess`** du constructeur.

`Read`	Accès en lecture seule au fichier.
`ReadWrite`	Accès en lecture et en écriture au fichier.
`Write`	Accès en écriture seule au fichier.

Tableau récapitulatif des associations permises du mode d'ouverture et du type d'accès

	Si le fichier existe			Si le fichier n'existe pas		
	`Read`	**`ReadWrite`**	**`Write`**	**`Read`**	**`ReadWrite`**	**`Write`**
`Append`	Err.	Err.	OK	Err.	Err.	OK
`Create`	Err.	OK	OK	Err.	OK	OK
`CreateNew`	Err.	Err.	Err.	Err.	OK	OK
`Open`	OK	OK	OK	Err.	Err.	Err.
`OpenOrCreate`	OK	OK	OK	OK	OK	OK
`Truncate`	Err.	OK	OK	Err.	Err.	Err.

Quelques membres de la classe `FileStream`

`Length` Cette propriété contient la taille du fichier en octets.

`Position` Cette propriété contient la position du prochain octet accessible dans le fichier. A l'ouverture du fichier, **`Position`** vaut **`0`**. Quand la fin de fichier est atteinte, **`Position`** a la même valeur que **`Length`**.

Close La méthode **Close** force l'enregistrement des données non encore transférées vers la mémoire de masse, ferme le fichier, et libères les ressources utilisées pour son exploitation.

Flush La méthode **Flush** force l'enregistrement des données non encore transférées vers la mémoire de masse et vide le buffer.

Seek Cette méthode force le déplacement du pointeur du fichier d'un nombre d'octets spécifié par rapport à une position également spécifiée et choisie parmi **SeekOrigin.Begin**, **SeekOrigin.Current** et **SeekOrigin.End**.

WriteByte Cette méthode écrit un seul **Byte** dans le fichier : **Fichier.WriteByte(UneVarDeTypeByte)**.

ReadByte Cette méthode lit un seul **Byte** du fichier : **Fichier.ReadByte(UneVarDeTypeByte)**.

L'écriture et la lecture du fichier

Outre les méthodes *marginales* **WriteByte** et **ReadByte**, ce sont les méthodes **Write** et **Read** qui permettent l'exploitation d'un fichier à accès direct. Elles ont en commun la manipulation d'un nombre donné de **Byte** appartenant ou affectés à un vecteur à partir d'un indice également donné. Cet indice peut avoir une des valeurs de **0** à **Count - 1**. Le nombre d'octets à lire ou écrire ne peut être supérieur à **Count – IndiceDepart**.

```
Fichier.Write(VecteurDeBytes, IndiceDansLeVecteur, NombreDeBytesAEcrire)
Fichier.Read(VecteurDeBytes, IndiceDansLeVecteur, NombreDeBytesALire)
```

La méthode **Read** retourne le nombre d'octets effectivement lus. Ce nombre peut être inférieur au **NombreDeBytesALire** demandé lorsque que la fin du fichier est atteinte.

```
Nombre = Fichier.Read(VecteurDeBytes, IndiceDansLeVecteur, NombreDeBytesALire)
```

L'usage d'un vecteur de **Byte** impose que les données à enregistrer soient converties dans ce type et que les données lues subissent la conversion inverse. La classe **System.Text.Encoding** offre le nécesaire à ces conversions sous la forme du jeu de propriétés énumérées ci-dessous et des méthodes **GetBytes** et **GetString**.

ASCII	Codage pour le jeu de caractères ASCII (7 bits).
Default	Codage de la page de codes ANSI actuelle du système.
UTF7	Codage du format UTF-7.
UTF8	Codage du format UTF-8.
Unicode	Codage du format UTF-16 avec primauté des octets de poids faible (Little endian).
BigEndianUnicode	Codage du format UTF-16 avec primauté des octets de poids fort (Big endian).
UTF32	Codage du format UTF-32 avec primauté des octets de poids faible (Little endian).

Le jeu de caractères ASCII sur 7 bits ne concerne que les caractères 0 à 127 et le jeu par défaut peut handicaper l'usage d'un fichier sur un autre système dont le format par défaut serait différent. L'UTF représente un des formats de codage du jeu de caractères universel (UTF pour **U**CS **T**ransformation **F**ormat et UCS pour **U**niversal **C**haracters **S**et). Il faut que le format choisi soit le même à l'enregistrement et à la lecture.

```
MonVecteur = System.Text.Encoding.UTF8.GetBytes(UneChaine)
UneChaine = System.Text.Encoding.UTF8.GetString(MonVecteur)
```

Les enregistrements doivent être d'une même longueur. Cela implique que les différents champs soient traités avant leur concaténation et leur conversion en tableau de **Byte**. Toutes les valeurs numériques peuvent aisément être converties en chaînes par **CType** ou **ToString** et toutes les chaînes doivent être ajustées à une longueur convenable.
Les méthodes **PadRight** et **PadLeft** de la classe **String** servent à compléter une chaîne en reproduisant un caractère donné jusqu'à l'obtention de la longueur souhaitée. De plus la chaîne traitée est alignée à droite ou à gauche selon la méthode choisie. La méthode **SubString** de la classe **String** permet de tronquer les chaînes trop longues. Les caractères excédentaires peuvent être enlevés à droite ou à gauche selon le paramétrage de la méthode.

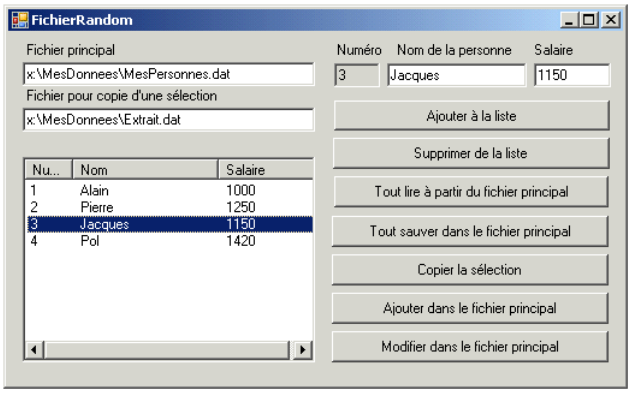
```
S = UneChaine.PadRight(6,"+"c)                    ' Si UneChaine est 1234,
                                                  ' le résulat S vaut 1234++
```

```
S = UneChaine.PadLeft(6,"~"c)                          ' Si UneChaine est 1234,
                                                       ' le résulat S vaut ~~1234
S = UneChaine.Substring(0, 4)                          ' Si UneChaine est 123456,
                                                       ' le résulat S vaut 1234
S = UneChaine.Substring(UneChaine.Length - 4, 4)       ' Si UneChaine est 123456,
                                                       ' le résulat S vaut 3456
```

Lors de la lecture des enregistrements, les méthodes **TrimStart** et **TrimEnd** de la classe **String** permettent la suppression aisée des caractères ajoutés. La méthode **Trim** sans paramètre ôte les espaces à gauche et à droite.

```
S = UneChaine.TrimStart("~")                           ' Si UneChaine est ~~1234,
                                                       ' le résulat S vaut 1234
S = UneChaine.TrimEnd("+")                             ' Si UneChaine est 1234++,
                                                       ' le résulat S vaut 1234
S = UneChaine.TrimStart()                              ' Enlève les espaces à gauche
S = UneChaine.TrimEnd()                                ' Enlève les espaces à droite
```

Exemple complet



L'application permet une gestion simple de personnes à l'écran par des boîtes de textes **TNom** et **TSalaire** destiné à l'encodage, ainsi qu'un **ListView LVPers** destiné à la consultation et aux sélections. L'information *Numéro* présente dans **TNumero** et dans la première colonne de **LVPers** ne peut être modifiée. Elle représente le numéro de l'enregistrement de la personne dans le fichier principal. Ce numéro vaut **0** tant que les données n'ont pas été effectivement enregistrées.
Les boîtes de textes **TFichierPrincipal** et **TFichierCopie** doivent recevoir les chemins et noms complets des fichiers. Le premier fichier est celui de toutes les personnes et le second est destiné à recevoir la copie d'enregistrements sélectionnés dans **LVPers**.
Les boutons *Ajouter à la liste* (**BAjouterListe**) et *Supprimer de la liste* (**BSupprimerListe**) permettent le pilotage de **LVPers**.
Les boutons *Tout lire …* (**BChargerTout**) et *Tout sauver …* (**BEnregistrerTout**) assurent respectivement la lecture et l'enregistrement de toutes les personnes du fichier principal.
Le bouton *Ajouter dans le fichier …* (**BAjouterFichier**) provoque l'écriture d'un seul enregistrement, celui dont les données sont présentes dans les **TextBox**. Le **ListView** est également mis à jour.
Le bouton *Modifier dans le fichier …* (**BModifierFichier**) provoque la ré-écriture de l'enregistrement dont le numéro et les données sont présentes dans les **TextBox**. Le **ListView** est également mis à jour.
Le bouton *Copier la sélection* (**BCopierSelection**) provoque l'enregistrement dans le deuxième fichier des enregistrements dont les données sont sélectionnées dans **LVPers**. Afin d'illustrer l'ouverture simultanée de deux fichiers et l'accès direct, chaque enregistrement envoyé vers le fichier de copie est d'abord lu dans le fichier principal sur base de son numéro présent dans **LVPers**.

```
Imports System.IO
Public Class FichierRandom

  Private Structure Personne                  ' La structure des données d'une personne
    Dim Nom As String                         ' associée aux réglages des longueurs des champs
    Dim Salaire As Integer                    ' ci-dessous, constitue la définition d'un
  End Structure                               ' enregistrement. Elle est écrite ici à titre
                                              ' d'exemple et n'est utilisée dans le code que
  Private LongNom As Integer = 30             ' lors de la lecture d'un fichhier complet
  Private LongSalaire As Byte = 6

  Private LongEnregistrement As Integer       ' Valeur calculée au démarrage du programme

  Private VBytes() As Byte                    ' L'enregistrement sous forme d'un vecteur de Byte
  Private UnEnregistrement As String          ' et sous forme d'une chaîne

  Private NomPers As String                   ' Les variables de manipulations des champs d'un
  Private SalPers As String                   ' enregistrement
```

```
' Programmation de quelques outils

' La fonction ByteVersString reconstitue une chaîne après la lecture d'un enregistrement

  Private Function BytesVersString(ByRef DesBytes() As Byte) As String
    Return System.Text.Encoding.UTF8.GetString(DesBytes)
  End Function

' La fonction StringVersByte crée un vecteur de Byte à partir d'une chaîne de données

  Private Function StringVersBytes(ByVal UneChaine As String) As Byte()
    Return System.Text.Encoding.UTF8.GetBytes(UneChaine)
  End Function

' La fonction ALongueur permet la mise à longueur et l'alignement des données des champs

  Private Function ALongueur(ByVal UneChaine As String, ByVal UneLongueur As Integer,
         Optional ByVal Alignement As Byte = 0, Optional ByVal Caractere As Char = " "c)
                                                                          As String
    Select Case Alignement
      Case 0                                ' Alignement à gauche (coupure ou ajout à droite)
        If UneChaine.Length > UneLongueur Then
          Return UneChaine.Substring(0, UneLongueur)
        Else
          Return UneChaine.PadRight(UneLongueur, Caractere)
        End If
      Case 1                                ' Alignement à droite (coupure ou ajout à gauche)
        If UneChaine.Length > UneLongueur Then
          Return UneChaine.Substring(UneChaine.Length - UneLongueur, UneLongueur)
        Else
          Return UneChaine.PadLeft(UneLongueur, Caractere)
        End If
    End Select
  End Function

' La fonction AccesAuFichier regroupe toutes les commandes d'ouverture et de fermeture des
' fichiers. Elle gère plusieurs fichiers par mise à jour d'un vecteur des références allouées
' à la demande. La fermeture d'un fichier libère sa référence qui peut être utilisée ensuite
' pour un autre fichier. Seules les associations Mode_Ouverture – Type_Accès utiles à cet exemple
' ont été programmées (Cas 1 à 4). Le bloc Select peut être complété selon les besoins.

  Private Function AccesAuFichier(ByVal Job As Byte, ByRef Canal As Byte,
                            Optional ByRef NomFichier As String = Nothing) As FileStream
    Static Dim Fichier() As FileStream = Nothing
    Static Dim CanalLibre As Integer = 0
    Dim NumCanal As Integer
    If Job < 9 Then                         ' Le Job 9 est réservé à la fermeture d'un fichier
      For NumCanal = 0 To CanalLibre – 1    ' Recherche d'un Fichier() non attribué
        If Fichier(NumCanal) Is Nothing Then
          Exit For
        End If
      Next
      If NumCanal < CanalLibre Then
        Canal = NumCanal                    ' Renvoyer l'indice par Canal passé ByRef
      Else
        Canal = CanalLibre                  ' S'il n'y a pas de référence libre dans Fichier(),
        ReDim Preserve Fichier(Canal)       ' il faut réallouer le vecteur en conséquence et
        Fichier(Canal) = Nothing            ' initialiser la nouvelle référence.
        CanalLibre += 1                     ' Préparation pour la prochaine réallocation.
      End If
    End If
    Select Case Job
      Case 9            ' Fermeture du fichier
        If Fichier(Canal) IsNot Nothing Then
          Fichier(Canal).Close()
          Fichier(Canal) = Nothing          ' La référence est réinitialisée après fermeture
        End If
```

```
      Case 1            ' Création pour écriture avec écrasement d'un existant éventuel
        Fichier(Canal) = New FileStream(NomFichier, FileMode.Create, FileAccess.Write)
      Case 2            ' Création ou ouverture pour écriture en ajout d'un existant éventuel
        Fichier(Canal) = New FileStream(NomFichier, FileMode.Append, FileAccess.Write)
      Case 3            ' Ouverture pour lecture d'un existant
        Fichier(Canal) = New FileStream(NomFichier, FileMode.Open, FileAccess.Read)
      Case 4            ' Ouverture pour lecture et écriture d'un existant
        Fichier(Canal) = New FileStream(NomFichier, FileMode.Open,FileAccess.ReadWrite)
    End Select
    Return Fichier(Canal)                                   ' Renvoi du FileStream ou
  End Function                                              ' Nothing si fichier fermé.

' Les procédures événementielles préalables

' Initialisations au démarrage du programme

  Private Sub FichierRandom_load(ByVal sender As Object, ByVal e As System.EventArgs)
                                                                        Handles Me.Load
    LongEnregistrement = LongNom + LongSalaire              ' Déterminer la longueur d'un
    ReDim VBytes(LongEnregistrement)                        ' enregistrement et allouer le
    LVPers.View = View.Details                              ' vecteur en conséquence.
    LVPers.FullRowSelect = True                             ' Réglages de la ListView
    LVPers.Columns.Add("Numéro", LVPers.Width \ 6, HorizontalAlignment.Left)
    LVPers.Columns.Add("Nom", LVPers.Width \ 2, HorizontalAlignment.Left)
    LVPers.Columns.Add("Salaire", LVPers.Width \ 3, HorizontalAlignment.Left)
    TNumero.ReadOnly = True                                 ' Interdire la modification
  End Sub                                                   ' du numéro d'enregistrement.

' Placer les données d'une ligne sélectionnée de LVPers dans les TextBox

  Private Sub LVPers_Click(ByVal sender As Object, ByVal e As System.EventArgs)
                                                                    Handles LVPers.Click
    TNumero.Text = LVPers.FocusedItem.Text
    TNom.Text = LVPers.FocusedItem.SubItems(1).Text
    TSalaire.Text = LVPers.FocusedItem.SubItems(2).Text
  End Sub

' Les procédures de traitements des données et des fichiers

  Private Sub BAjouterListe_Click(ByVal sender As Object, ByVal e As System.EventArgs)
                                                          Handles BAjouterListe.Click
    Dim LV As ListViewItem
    If TNom.Text = "" Or TSalaire.Text = "" Then Exit Sub   ' Données non valides
    LV = LVPers.Items.Add("0")                              ' Numéro 0 car non enregistré
    LV.SubItems.Add(TNom.Text)
    LV.SubItems.Add(TSalaire.Text)
  End Sub

  Private Sub BSupprimerListe_Click(ByVal sender As System.Object,
                            ByVal e As System.EventArgs) Handles BSupprimerListe.Click
    Dim NrLigne As Integer
    If TNom.Text = "" Or TSalaire.Text = "" Then Exit Sub   ' Données non valides
    For NrLigne = 0 To LVPers.Items.Count - 1
      If LVPers.Items(NrLigne).Text = TNumero.Text Then     ' Recherche sur numéro et nom
        If LVPers.Items(NrLigne).SubItems(1).Text = TNom.Text Then
          LVPers.Items.RemoveAt(NrLigne)
          TNumero.Clear()
          TNom.Clear()
          TSalaire.Clear()
          Exit For
        End If
      End If
    Next
  End Sub
```

```
Private Sub BAjouterFichier_Click(ByVal sender As Object,
                                  ByVal e As System.EventArgs) Handles BAjouterFichier.Click
    Dim FAjoute As FileStream
    Dim NumeroCanal As Byte
    Dim LV As ListViewItem
    If TNom.Text = "" Or TSalaire.Text = "" Then Exit Sub  ' Données non valides
    If TFichierPrincipal.Text = String.Empty Then          ' Nom de fichier requis
      MessageBox.Show("Désignez un fichier principal.")
      TFichierPrincipal.Select()
      Exit Sub
    End If
    If TNumero.Text >= "1" Then           ' Cette personne existe déjà dans le fichier
      BModifierFichier_Click(Me, Nothing)
      Exit Sub
    End If

    ' Ouvrir le fichier en création ou ajout et conserver le NumeroCanal pour sa fermeture

    FAjoute = AccesAuFichier(2, NumeroCanal, TFichierPrincipal.Text)

    ' Mettre les champs à longueur, créer la chaînes des données, la convertir en vecteur de Byte

    NomPers = ALongueur(TNom.Text, LongNom, 0)             ' Aligner à gauche
    SalPers = ALongueur(TSalaire.Text, LongSalaire, 1)     ' Aligner à droite
    UnEnregistrement = NomPers & SalPers
    VBytes = StringVersBytes(UnEnregistrement)

    ' Enregistrer le vecteur de Byte

    FAjoute.Write(VBytes, 0, LongEnregistrement)

    ' Mettre à jour LVPers. Le numéro de l'enregistrement est calculé à partir de Position

    LV = LVPers.Items.Add(FAjoute.Position / LongEnregistrement)
    LV.SubItems.Add(TNom.Text)
    LV.SubItems.Add(TSalaire.Text)

    ' Fermer le fichier

    AccesAuFichier(9, NumeroCanal)
  End Sub

  Private Sub BModifierFichier_Click(ByVal sender As System.Object,
                                     ByVal e As System.EventArgs) Handles BModifierFichier.Click
    Dim FModifie As FileStream
    Dim NumeroCanal As Byte
    Dim NrLigne As Integer
    If TNom.Text = "" Or TSalaire.Text = "" Then Exit Sub  ' Données non valides
    If TFichierPrincipal.Text = String.Empty Then
      MessageBox.Show("Désignez un fichier principal.")
      TFichierPrincipal.Select()
      Exit Sub
    End If
    If TNumero.Text < "1" Then            ' Cette personne n'existe pas dans le fichier
      BAjouterFichier_Click(Me, Nothing)
      Exit Sub
    End If

    ' Ouvrir le fichier en lecture-écriture et conserver le NumeroCanal pour sa fermeture

    FModifie = AccesAuFichier(4, NumeroCanal, TFichierPrincipal.Text)

    ' Mettre les champs à longueur, créer la chaînes des données, la convertir en vecteur de Byte

    NomPers = ALongueur(TNom.Text, LongNom, 0)             ' Aligner à gauche
    SalPers = ALongueur(TSalaire.Text, LongSalaire, 1)     ' Aligner à droite
    UnEnregistrement = NomPers & SalPers
    VBytes = StringVersBytes(UnEnregistrement)
```

```
  ' Positionner le pointeur sur l'enregistrement dont le numéro est donné par TNumero

 FModifie.Seek((CType(TNumero.Text, Integer) - 1) * LongEnregistrement,
                                                                    SeekOrigin.Begin)
  ' Enregistrer le vecteur de Byte

 FModifie.Write(VBytes, 0, LongEnregistrement)

  ' Mettre à jour LVPers là où le numéro d'enregistrement est le même que dans TNumero

  For NrLigne = 0 To LVPers.Items.Count – 1
    If LVPers.Items(NrLigne).Text = TNumero.Text Then
      LVPers.Items(NrLigne).SubItems(1).Text = TNom.Text
      LVPers.Items(NrLigne).SubItems(2).Text = TSalaire.Text
      Exit For
    End If
  Next

  ' Fermer le fichier

 AccesAuFichier(9, NumeroCanal)
End Sub

Private Sub BChargerTout_Click(ByVal sender As Object, ByVal e As System.EventArgs)
                                                          Handles BChargerTout.Click
  Dim FEntree As FileStream
  Dim NumeroCanal As Byte
  Dim UnePersonne As Personne
  Dim LV As ListViewItem
  If TFichierPrincipal.Text = String.Empty Then          ' Nom de fichier requis
    MessageBox.Show("Désignez un fichier principal.")
    TFichierPrincipal.Select()
    Exit Sub
  End If
  LVPers.Items.Clear()

  ' Ouvrir le fichier en lecture seule et conserver le NumeroCanal pour sa fermeture

  FEntree = AccesAuFichier(3, NumeroCanal, TFichierPrincipal.Text)

  ' Lire les enregistrements de 1 à FEntree.Length / LongEnregistrement

  For NrEnreg As Integer = 1 To FEntree.Length / LongEnregistrement
    FEntree.Read(VBytes, 0, LongEnregistrement)

  ' Extraire les champs de chaque vecteur de Byte et supprimer les caractères de mise à
  ' longueur utilisés lors de l'enregistrement. Comme ce sont des espaces qui ont été utilisés,
  ' la méthode Trim sans paramètre suffit pour les ôter à gauche et à droite.

    UnEnregistrement = BytesVersString(VBytes)
    With UnePersonne
      .Nom = UnEnregistrement.Substring(0, LongNom).Trim
      .Salaire = CType(UnEnregistrement.Substring(LongNom, LongSalaire).Trim,
                                                                        Integer)
  ' Afficher les données lues dans LVPers

      LV = LVPers.Items.Add(NrEnreg)
      LV.SubItems.Add(.Nom)
      LV.SubItems.Add(.Salaire.ToString)
    End With
  Next NrEnreg

  ' Fermer le fichier

  AccesAuFichier(9, NumeroCanal)
End Sub
```

```
Private Sub BEnregistrerTout_Click(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles BEnregistrerTout.Click
  Dim FSortie As FileStream
  Dim NumeroCanal As Byte
  If TFichierPrincipal.Text = String.Empty Then          ' Nom de fichier requis
    MessageBox.Show("Désignez un fichier principal.")
    TFichierPrincipal.Select()
    Exit Sub
  End If

  ' Ouvrir le fichier en écrasement de l'existant et conserver le NumeroCanal pour sa fermeture

  FSortie = AccesAuFichier(1, NumeroCanal, TFichierPrincipal.Text)

  ' Lire les données dans LVPers (sans le numéro d'enregistrement)

  For NrLigne As Integer = 0 To LVPers.Items.Count - 1

  ' Mettre les champs à longueur, créer la chaînes des données, la convertir en vecteur de Byte

    NomPers = ALongueur(LVPers.Items(NrLigne).SubItems(1).Text, LongNom, 0)
    SalPers = ALongueur(LVPers.Items(NrLigne).SubItems(2).Text, LongSalaire, 1)
    UnEnregistrement = NomPers & SalPers
    VBytes = StringVersBytes(UnEnregistrement)

  ' Enregistrer le vecteur de Byte

   FSortie.Write(VBytes, 0, LongEnregistrement)
  Next

  ' Fermer le fichier

  AccesAuFichier(9, NumeroCanal)
End Sub

Private Sub BCopierSelection_Click(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles BCopierSelection.Click
  Dim FSource As FileStream
  Dim FCible As FileStream
  Dim NumeroCanalSource As Byte
  Dim NumeroCanalCible As Byte                           ' Nom de fichier requis et il
  If TFichierCopie.Text = TFichierPrincipal.Text Or      ' doit être différent du source
                                                TFichierCopie.Text = String.Empty Then
    MessageBox.Show("Désignez un fichier différent du principal.")
    TFichierCopie.Select()
    Exit Sub
  End If

  ' Ouvrir le fichier source en lecture seule et conserver NumeroCanalSource pour sa fermeture
  ' et ouvrir le fichier cible en écrasement d'un existant éventuel et conserver
  ' NumeroCanalCible pour sa fermeture

  FSource = AccesAuFichier(3, NumeroCanalSource, TFichierPrincipal.Text)
  FCible = AccesAuFichier(1, NumeroCanalCible, TFichierCopie.Text)

  ' Rechercher les lignes sélectionnées dans LVPers

  For NrLigne As Integer = 0 To LVPers.Items.Count - 1
    If LVPers.Items(NrLigne).Selected Then
      If LVPers.Items(NrLigne).Text > "0" Then

  ' Un numéro d'enregistrement supérieur à 0 dans LVPers, signifie que l'enregistrement existe
  ' dans le fichier source. Il est alors lu après positionnement direct dans le fichier source
  ' et le vecteur conservé tel quel.

        FSource.Seek((CType(LVPers.Items(NrLigne).Text, Integer) - 1)
                                              * LongEnregistrement, SeekOrigin.Begin)
        FSource.Read(VBytes, 0, LongEnregistrement)
      Else
```

```
    ' Un numéro d'enregistrement égal à 0 dans LVPers, signifie que les données de la ligne
    ' n'ont pas encore été enregistrées dans le fichier principal et il n'est donc pas possible
    ' de les relire. Dans ce cas, il faut mettre les champs à longueur, créer la chaînes des
    ' données, la convertir en vecteur de Byte

          NomPers = ALongueur(LVPers.Items(NrLigne).SubItems(1).Text, LongNom, 0)
          SalPers = ALongueur(LVPers.Items(NrLigne).SubItems(2).Text, LongSalaire, 1)
          UnEnregistrement = NomPers & SalPers
          VBytes = StringVersBytes(UnEnregistrement)
        End If

    ' Enregistrer le vecteur de Byte, lu ou créé, dans le fichier cible

        FCible.Write(VBytes, 0, LongEnregistrement)
      End If
    Next

    ' Fermer les fichiers

    AccesAuFichier(9, NumeroCanalCible)
    AccesAuFichier(9, NumeroCanalSource)
  End Sub
End Class
```

La protection des données des fichiers lors des accès simultanés

La protection des données des fichiers lors des accès simultanés se réalise par l'emploi de la méthode **`Lock`** de la classe **`FileStream`**. La méthode **`Lock`** permet le verrouillage d'un ou plusieurs enregistrements (*Record Locking*), voire même de tout un fichier (*File Locking*). Le verrouillage limité aux seuls enregistrements concernés à un moment donné, présente l'avantage de préserver l'accès aux autres enregistrements par les utilisateurs qui peuvent ainsi continuer à travailler sans perte de temps. La libération d'enregistrements verrouillés se commande par la méthode **`UnLock`**, avec les mêmes paramètres que ceux employés lors du verrouillage :

```
Dim Fichier As FileStream
Dim Position As Long
Dim NombreOctet As Long
' … … …
Fichier.Lock(Position, NombreOctet)
' … … … personne d'autre que moi ne peut modifier ces octets …
Fichier.UnLock(Position, NombreOctet)
```

La protection des données des fichiers par l'usage de ces méthodes consiste à s'assurer qu'une donnée en cours de mise à jour, n'est pas modifiée dans le même temps par un autre utilisateur ou par une autre instance de l'application. Les données en mémoire sont d'abord copiées dans des variables temporaires avant d'être modifiées et les valeurs contenues dans le fichier sont relues avant l'enregistrement des modifications. L'enregistrement lu est verrouillé avant la relecture et n'est libéré qu'après l'enregistrement ou l'abandon des modifications.

Voici le procédé :

- Copie des valeurs sur le point d'être modifiées dans des variables temporaires (**`Tmp1`**).
- Modification des valeurs des variables de travail.
- Verrouillage de l'enregistrement dans le fichier.
- Relecture de l'enregistrement à modifier dans des variables temporaires (**`Tmp2`**).
- Comparaison des variables temporaires (**`Tmp1`** et **`Tmp2`**).
- Si les variables temporaires sont égales alors
  mettre le fichier à jour avec les variables de travail modifiées
  sinon
  traiter la situation : les données ont été modifiées entre temps
- Déverrouiller l'enregistrement après la mise à jour réussie ou après abandon des modifications

La procédure **`BModifierFichier_Click`** de l'application précédente est récrite ci-après à titre d'illustration de ce procédé. Son expérimentation se réalise en instanciant deux fois la même application, chaque instance traitant le même enregistrement du même fichier.

```
Private Sub BModifierFichier_Click(ByVal sender As System.Object,
                                   ByVal e As System.EventArgs) Handles BModifierFichier.Click
  Dim FModifie As FileStream
  Dim NumeroCanal As Byte
  Dim NrLigne As Integer
  Dim TmpNom1, TmpSal1 As String        ' Déclaration d'un jeu de variables temporaires
  Dim TmpNom2, TmpSal2 As String        ' Déclaration de l'autre jeu
  Dim OKModif As Boolean
          ' … … …
          ' Voir les quelques contrôles de validité dans l'exemple précédent
          ' … … …
  ' Ouvrir le fichier en lecture-écriture et conserver le NumeroCanal pour sa fermeture

  FModifie = AccesAuFichier(4, NumeroCanal, TFichierPrincipal.Text)

  ' Stockage des infos telles qu'elles ont été lues précédemment dans le fichier
  For NrLigne = 0 To LVPers.Items.Count - 1
    If LVPers.Items(NrLigne).Text = TNumero.Text Then
      TmpNom1 = LVPers.Items(NrLigne).SubItems(1).Text
      TmpSal1 = LVPers.Items(NrLigne).SubItems(2).Text
      Exit For
    End If
  Next

  ' Relecture des infos actuelles du fichier après verrouillage de l'enregistrement
  FModifie.Lock((CType(TNumero.Text, Integer) - 1) * LongEnregistrement,
                                                          LongEnregistrement)
  FModifie.Seek((CType(TNumero.Text, Integer) - 1) * LongEnregistrement,
                                                            SeekOrigin.Begin)
  FModifie.Read(TByte, 0, LongEnregistrement)
  TmpNom2 = BytesVersString(TByte).Substring(0, LongNom).Trim
  TmpSal2 = CType(BytesVersString(TByte).Substring(LongNom, LongSalaire).Trim,
                                                                  Integer)
  ' Vérifier si des modifications ont été faites entre temps
  If TmpNom1 <> TmpNom2 Or TmpSal1 <> TmpSal2 Then
    If MessageBox.Show("Enregistrement modifié entre temps. Continuer ? ",
           "Attention", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then
      TNom.Text = TmpNom2         ' Mettre à jour nos données avec celles du fichier
      TSalaire.Text = TmpSal2
      OKModif = False
    Else
      OKModif = True              ' ou remplacer les données du fichier avec les nôtres
    End If
  Else
    OKModif = True                ' Le fichier n'a pas été modifié entre temps
  End If

  If OKModif Then
  ' Mettre les champs à longueur, créer la chaîne des données, la convertir en vecteur de Byte
          ' … … …
          ' Voir les quatres lignes de codes dans l'exemple précédent
          ' … … …

  ' Positionner le pointeur sur l'enregistrement est donné par TNumero
    FModifie.Seek((CType(TNumero.Text, Integer) - 1) * LongEnregistrement,
                                                            SeekOrigin.Begin)
  ' Enregistrer le vecteur de Byte
    FModifie.Write(VBytes, 0, LongEnregistrement)
  End If

  ' Déverrouiller l'enregistrement
  FModifie.Unlock((CType(TNumero.Text, Integer) - 1) * LongEnregistrement,
                                                          LongEnregistrement)
  ' Mettre à jour LVPers là où le numéro d'enregistrement est le même que dans TNumero
          ' … … …
          ' Voir le reste de la procédure dans l'exemple précédent
          ' … … …
End Sub
```