Cours Apl 12 - La gestion des fichiers + quelques autres notions nouvelles

Dyalog Apl vous permet de manipuler 2 principaux types de fichiers :
- les fichiers dits natifs ou textes
- les fichiers Apl ou fichiers à composantes

Quel que soit le type de fichier, les étapes principales sont toujours les mêmes :
- si le fichier existe, on s'y lie en indiquant son chemin et on crée par la même occasion un numéro de lien, utilisé pendant toute la durée de la manoeuvre.
- si on crée un nouveau fichier, on indique le chemin et le fichier est créé, la fonction de création rendant un numéro de lien.
- Au moyen du numéro de lien, on peut lire et écrire dans le fichier ouvert
- Quand on a fini, on le libère.

* Les fichiers texte :

Toutes les fonctions de manipulation de fichiers textes commencent par ⎕*N*, le N signifiant "Native" : natif en anglais.

- Création de fichier natif: `r←g ⎕NCreate d`
g : chemin et nom du nouveau fichier
d : numéro de lien à utiliser. Il est plus simple de laisser Apl choisir un numéro lui-même, dans ce cas, on met zéro comme numéro de lien
r : numéro de lien. Égal à d si on a mis autre chose que zéro.

Exemple : créez le fichier zouzou.txt dans le même répertoire que votre ws de TD.
- Cherchons d'abord le chemin de notre workspace.
La fonction ⎕*WS*id rend une chaîne contenant le chemin et le nom du ws.
Pour récupérer uniquement le chemin, nous appliquons l'algorithme suivant :
- on découpe le chemin/nom du ws en vecteur de vecteurs dont chacun commence par un \
- on abandonne le dernier (\nom du ws)
- on concatène les éléments restants en une chaîne unique
- on ajoute \ à la fin de cette chaîne

Nous allons utiliser pour cela le ⊂ **en mode "split"**.

Il admet en argument gauche un vecteur booléen pilotant le découpage.
Il est de même dimension que le vecteur à découper.
Essayez ceci :

```
  DISPLAY 1 0 0 1 0 1⊂'abcdef'
|→----------------|
| |→--| |→-| |→| |
| |abc| |de| |f| |
| |---| |--| |-| |
|∈---------------|
```

Pour découper, notre but est d'avoir un 1 à chaque fois qu'un \ est rencontré :
```
(1,1↓⎕WSID='\')
```
On remplace systématiquement le premier booléen par un 1 afin que la partie de chaîne à gauche du premier '\' soit bien conservée.

Pour connaître le chemin nous écrirons donc :
```
Rep←(⊃,/¯1↓(1,1↓⎕WSid='\')⊂⎕WSid),'\'
```

- on découpe le chemin/nom du ws en vecteur de vecteurs dont chacun commence par un \ :
```
(1,1↓⎕WSid='\')⊂⎕WSid
```
- on abandonne le dernier (\nom du ws) :
```
¯1↓
```
- on concatène les éléments restants (réduction par concaténation) :
```
,/
```
- on extrait le vecteur de sa profondeur :
```
⊃
```
- on ajoute \ à la fin de cette chaîne
```
,'\'
```

Reprenons la création du fichier zouzou.txt
```
Numf←(Rep,'zouzou.txt')⎕Ncreate 0
```

Numf est le numéro de lien pour ce fichier.

- Pour connaître les numéros des fichiers ouverts :
```
⎕Nnums
```

- Pour connaître les noms des fichiers ouverts :
```
⎕Nnames
```

- Pour connaître les deux :
```
⎕Nnums, ⎕Nnames
```

- Ecriture dans un fichier natif :

- à la suite du contenu déjà existant :
```
vecteur_de_données ⎕Nappend num_lien
```

- à la place du contenu déjà existant :
```
vecteur_de_données ⎕Nreplace num_lien, emplacement du début d'écriture (numéro d'octet).
```

Pour marquer les fins de lignes (nouvelle ligne, retour chariot) il faut y concaténer les 2 caractères CR et LF.
On peut les obtenir de 2 manières :
- ⎕TC[3 2]
- ⎕AV[4 3]

⎕AV contient les 256 caractères utilisés par Apl.

Donc pour ajouter "une première ligne" à zouzou.txt, on procèdera ainsi :
```
('Une première ligne.', ⎕AV[4 3]) ⎕Nappend Numf
```

Encore une ligne ?
```
('Encore une ligne.', ⎕AV[4 3]) ⎕Nappend Numf
```

- Refermer un fichier natif :
```
⎕Nuntie Num_fichier
```

- Ouvrir un fichier déjà existant :
Les arguments sont les mêmes que pour la création de fichier :
```
NumLien ← chemin_et_nom_fichier ⎕Ntie numéro_lien_à_utiliser_ou_0
```

- Lecture dans un fichier natif :
```
Vecteur ← ⎕Nread Num_fichier, code conversion (en général 82), nombre de caractères à lire, position début de lecture
```

Le dernier argument est facultatif.
Concernant le nombre de caractères à lire, on indique généralement un très grand nombre afin d'être sûr de tout récupérer d'un coup.

Pour relire notre fichier :
```
zouzou ← ⎕Nread Numf, 82, 100000
```

⍴ de zouzou = 40

Si nous voulons le transformer en un vecteur de vecteurs dont chaque élément serait une ligne du fichier, nous devons découper le vecteur en fonction des fins de lignes (`⎕AV[4 3]`) et enlever ces séparateurs.

```
2↓¨(1 0,⎕AV[4 3]∊zouzou)⊂'  ',zouzou
```

- Petite astuce pour refermer tous les fichiers natifs :
```
⎕Nuntie ⎕Nnums
```

- Autres fonctions :
  - **renommer un fichier** : `⎕Nrename`
  - **effacer un fichier** : `⎕Nerase`
  - **taille d'un fichier** : `⎕Nsize`

* Les fichiers Apl ou fichiers à composantes

On peut se représenter ces fichiers comme des vecteurs généralisés dont chaque élément peut être le contenu d'une variable de type et de structure quelconque.

- Création de fichier à composante :

```
NumLien ← chemin_et_nom_fichier ⎕FCREATE numéro_lien_à_utiliser_ou_0
```

Exemple : créez le fichier zouzou2 dans le même répertoire que votre ws de TD.

```
Rep←(⊃,/¯1↓(1,1↓⎕WSid='\')⊂⎕WSid),'\'
Numf←(Rep,'zouzou2')⎕FCREATE 0
```

- Pour connaître les numéros des fichiers ouverts :
⎕*fn*ums
- Pour connaître les noms des fichiers ouverts :
⎕*fn*ames
- Pour connaître les deux :
⎕*fn*ums, ⎕*fn*ames

- Ecriture dans un fichier à composante :

- à la suite du contenu déjà existant : `objet_apl ⎕fappend num_lien`
- à la place du contenu déjà existant : `objet_apl ⎕freplace num_lien, num_composante`

Exemple : Nous allons écrire dans les 3 premières composantes de zouzou2 le contenu des matrices Mnum1, Mnum2 et Mnum3.

```
Mnum1 Mnum2 Mnum3 ⎕fappend¨Numf
```

On utilise le each pour éviter d'appeler 3 fois fappend.

Si on veut remplacer la troisième composante par le contenu de la variable Vnum1 :

```
Vnum1 ⎕freplace Numf 3
```

- Refermer un fichier à composantes :

```
⎕funtie Num_fichier
```

- Ouvrir un fichier déjà existant :

Les arguments sont les mêmes que pour la création de fichier :

```
NumLien ← chemin_et_nom_fichier  ⎕FTIE numéro_lien_à_utiliser_ou_0
```

Par contre, on peut ouvrir le fichier en mode exclusif `( ⎕FTIE)` ou partagé `(⎕fstie)`.

- Lecture dans un fichier à composantes :

La lecture de ce type de fichier est très simple puisqu'on récupère directement un objet Apl.

```
Variable ← ⎕fread Num_fichier, num_composante
```

- Autres fonctions :
  - renommer un fichier : `⎕frename`
  - effacer un fichier : `⎕ferase`
  - taille d'un fichier : `⎕fsize`

Remarques sur les fichiers à composantes :
Souvent la première composante fait office de sommaire, représentant de manière synthétique les composantes suivantes.

Par exemple, on pourrait dans une école avoir un fichier des classes avec en première composante un vecteur de la liste des classes du type :
"CP A", "CP B", "CE1 A", "CE1 B", etc ...

Ainsi on peut immédiatement afficher la liste des classes.
Par ailleurs, l'utilisation du iota dyadique permet instantanément de retrouver la position de la classe recherchée.
Le iota dyadique donne la position des éléments de son argument droit au sein de son argument gauche.
Exemple :

```
      1 3 'Louis' 2 'Yves' ⍳ 3 4 'Louis'
2 6 3
```

3 a été trouvé en 2ème position, 4 n'a pas été trouvé et Louis a été trouvé en 3ème position.
Si un des éléments de droite n'est pas trouvé à gauche, Apl donne sa position comme étant égale à la longueur de G + 1.

Ainsi pour récupérer la composante du CE1 A, on écrirait :

```
Classe ← ⎕fread Numfic, ((⎕fread Numfic 1) ⍳ ⊂ 'CE1 A') + 1
```

Les fichiers Apl sont également souvent organisés en fichiers dits "inversés".
En effet, on a souvent tendance à créer une composante par enregistrement, ce qui semble la tendance la plus naturelle.
En revanche, si on a beaucoup d'enregistrements et qu'on veut en sélectionner en fonction de certains critères, il faut alors tous les ouvrir pour voir si ils correspondent aux conditions.
Dans le cas d'un fichier inversé, chaque composante contient un champ de tous les enregistrements.
Par exemple dans le cas de notre TD, le fichier du personnel aurait 4 composantes :
- nom
- sexe
- age
- salaire annuel.

Travaux pratiques :

0. Chargez votre Ws de travaux pratiques :
)load c:\Mes documents\pratique-apl

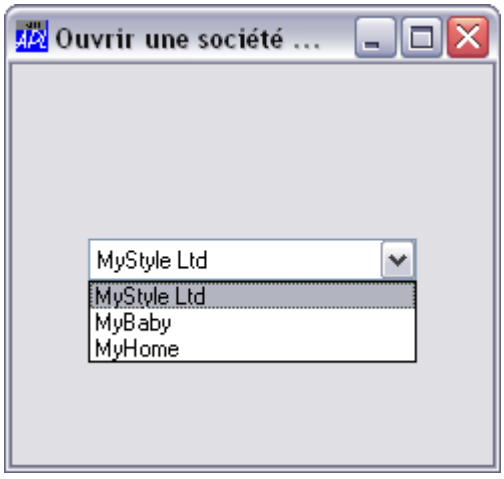
La société MyStyle Ltd s'est pas mal développée et elle a finalement créé 2 filiales : MyBaby, spécialisée dans les vêtements pour enfants et MyHome, spécialisée en design et mobilier d'intérieur.

Votre logiciel de gestion du personnel doit donc désormais être multi-sociétés.

1. Créez un fichier à composantes intitulé PersoGroup, dans le même répertoire que votre ws.
La première composante servira de sommaire.
Ce sera un vecteur de vecteur comportant les noms des sociétés gérées.
Les composantes suivantes contiendront les matrices de personnels correspondantes.
Dans un premier temps, mettez simplement "MyStyle Ltd" et "MyBaby" dans la première composante, le contenu de la variable Perso dans la deuxième et une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et de 2 zéros dans la troisième.

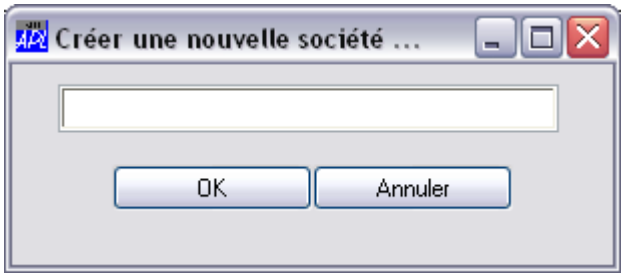
2. Modifiez GestPerso de manière à ce que la première société du fichier se charge automatiquement à l'ouverture. Pensez également à mettre à jour la propriété "Caption" de la fenêtre.
Conseil : Ecrivez une fonction "GestPersoLoad" admettant en argument droit le numéro de société à charger et ouvrant le fichier uniquement le temps de récupérer les données. Cette fonction se chargera également du rafraîchissement de l'écran.
Vous la réutiliserez plus tard.

3. Ajoutez un menu intitulé "Fichier" avec les éléments suivants :
- Ouvrir Société qui ouvre une nouvelle fenêtre avec une combo contenant la liste de toutes les sociétés. Une fois la société choisie, cet écran se referme et elle est chargée à l'écran via GestPersoLoad.



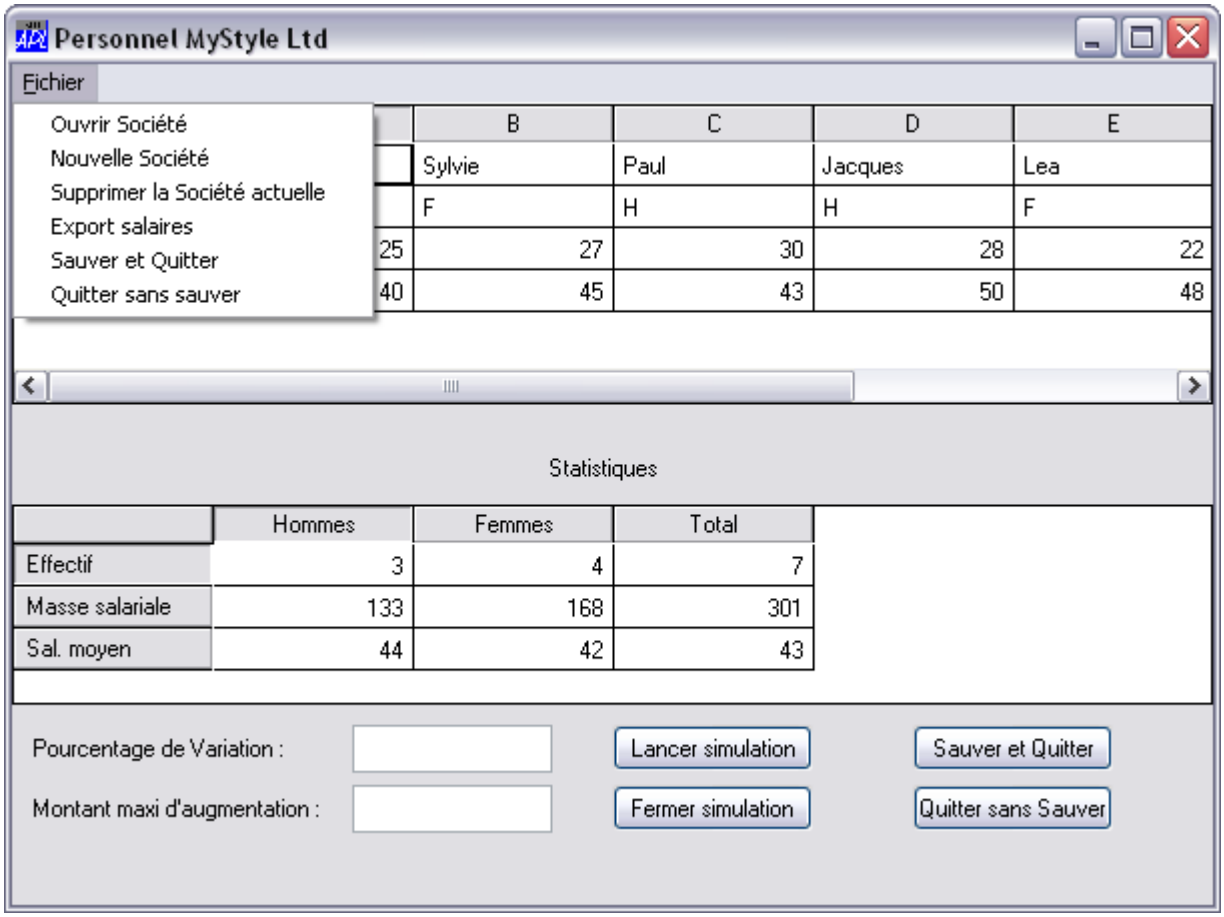
- Nouvelle Société
Ce choix provoque l'ouverture d'une nouvelle fenêtre avec
    - un champ pour saisir le nom de la nouvelle société
    - un bouton Ok
    - un bouton Annuler

Lorsque le nom de la société est saisi et validé on l'ajoute à la liste des sociétés dans la première composante, on crée une nouvelle composante contenant une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et 2 zéros et on affiche le tout à l'écran.



- Supprimer la Société actuelle
Si cette option est choisie, on commence par demander confirmation de la société en reprécisant bien son nom dans la question.
Si la suppression est confirmée, on opère les changements suivants sur le fichier :
- dans la première composante on met iota zéro (⍳0) à la place du nom de la société effacée.
- on remplace la matrice de la société supprimée par une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et 2 zéros.
Enfin, on affiche la première société à l'écran.

- Sauver et Quitter : Comme le bouton mais on sauve dans le fichier
- Quitter sans sauver : Comme le bouton.



4. A l'aide de GestPerso, saisissez le personnel de MyBaby :
- Tanguy, homme, 27 ans, 25 keuro/an
- Yvette, femme, 35 ans, 30 keuro/an
- Renée, femme, 50 ans, 28 keuro/an
- Simon, homme, 45 ans, 33 keuro/an

Sauvez puis créez la société MyHome et saisissez son personnel :
- Eric, homme, 37 ans, 35 keuro/an
- Isabelle, femme, 35 ans, 30 keuro/an
- Mishu, femme, 50 ans, 38 keuro/an
- Raoul, homme, 40 ans, 37 keuro/an

5. Le département comptabilité voudrait un fichier texte des noms et salaires pour chaque société.
Fichier sur 2 colonnes : nom;salaire
Exemple :
Pierre;40
Sylvie;45
Paul;43
Jacques;50

Ajoutez un menu "Export Salaires" juste avant "sauver et quitter" qui, au moyen d'un objet FileBox, permet de choisir précisément le nom et l'emplacement du fichier. Le nom proposé par défaut sera "Salaires société.txt"

6. Enregistrez votre travail avec)save.

Solutions :

1. Créez un fichier à composantes intitulé PersoGroup, dans le même répertoire que votre ws.

```
Numf←(Rep,'PersoGroup')⎕FCREATE 0
```

La première composante servira de sommaire.
Dans un premier temps, mettez simplement "MyStyle Ltd" et "MyBaby" dans la première composante :

```
'MyStyle Ltd' 'MyBaby' ⎕fappend Numf
```

, le contenu de la variable Perso dans la deuxième :

```
Perso ⎕fappend Numf
```

et une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et de 2 zéros dans la troisième.

```
(4 1⍴'' '' 0 0) ⎕fappend Numf
```

on ferme ensuite le fichier :

```
⎕funtie Numf
```

2. Modifiez GestPerso de manière à ce que la première société du fichier se charge automatiquement à l'ouverture. Pensez également à mettre à jour la propriété "Caption" de la fenêtre.
Conseil : Ecrivez une fonction "GestPersoLoad" admettant en argument droit le numéro de société à charger et ouvrant le fichier uniquement le temps de récupérer les données. Cette fonction se chargera également du rafraîchissement de l'écran.

On commence par localiser les variables `soc` (nom de société) et `chemin` (chemin du ws) dans la fonction GestPerso :

```
 GestPerso;F1;tmp;PersoLoc;soc;chemin
```

Toujours dans GestPerso, vers le début de la fonction, on renseigne la variable "chemin" :

```
⍝ Chemin du ws
 chemin←(⊃,/¯1↓(1,1↓⎕WSID=''\'')⊂⎕WSID),''\''
```

Texte de GestPersoLoad :

```
 GestPersoLoad d;Numf
⍝ Charger la société numéro d
 Numf←(chemin,'PersoGroup')⎕FTIE 0
 soc←d⊃⎕fread Numf 1
 F1.Caption←'Personnel ',soc
 Perso←⎕fread Numf(d+1)
 F1.G1.Values←Perso
 F1.G1.CellTypes←(⍴F1.G1.Values)⍴3
 F1.G1.CellTypes[1;]←1
 F1.G1.CellTypes[2;]←2
 GestPerso_Stats 1
 PersoLoc←Perso
 ⎕FUNTIE Numf
```

Cette fonction reprend la plupart des lignes de GestPerso puisqu'elle écrase après coup certaines propriétés des objets tels que les valeurs du grid principal ou les types de cellules.
On appellera cette fonction juste avant l'activation de l'écran (`⎕DQ`) en chargeant la première société du fichier :

```
GestPersoLoad 1
```

3. Ajoutez un menu intitulé "Fichier" avec les éléments suivants :
- Ouvrir Société qui ouvre une nouvelle fenêtre avec une combo contenant la liste de toutes les sociétés. Une fois la société choisie, cet écran se referme et elle est chargée à l'écran via GestPersoLoad.

Nouvelles lignes dans GestPerso :

```
 'F1.MB'⎕WC'MenuBar'
 'F1.MB.M1'⎕WC'Menu' '&Fichier'
 'F1.MB.M1.OS'⎕WC'MenuItem' '&Ouvrir Société'('Event' 'Select'
'GestPersoOpen')
```

Texte de la fonction GestPersoOpen :

```
GestPersoOpen;Numf;FS;ListSoc;res;ls2
⍝ Ouvrir une des sociétés de la base
 Numf←(chemin,'PersoGroup')⎕FTIE 0
 ls2←ListSoc←⎕FREAD Numf 1
 'FS'⎕WC'Form' 'Ouvrir une société ...'('Coord' 'Pixel')('Size' 200 240)
 'FS.C1'⎕WC'Combo'('Items'ls2)('event' 'select' 1)
 FS.C1.SelItems←ls2¨⊂soc
 ⎕FUNTIE Numf
 res←⎕DQ'FS'
 →(0=⍴res)/0
 →('FS.C1'≠1⊃res)/0
 GestPersoLoad ListSoc⍳⊂FS.C1.Text
```

- Nouvelle Société
Ce choix provoque l'ouverture d'une nouvelle fenêtre avec
- un champ pour saisir le nom de la nouvelle société
- un bouton Ok
- un bouton Annuler

Lorsque le nom de la société est saisi et validé on l'ajoute à la liste des sociétés dans la première composante, on crée une nouvelle composante contenant une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et 2 zéros et on affiche le tout à l'écran.

Nouvelle ligne de menu dans GestPerso :

```
'F1.MB.M1.NS'⎕WC'MenuItem' '&Nouvelle Société'('Event' 'Select'
'GestPersoNew')
```

Texte de la fonction GestPersoNew :

```
GestPersoNew;Numf;FS;ListSoc;res;msg;place
⍝ Créer une nouvelle société dans la base
 'FS'⎕WC'Form' 'Créer une nouvelle société ...'('Coord' 'Pixel')('Size'
100 300)
 'FS.E1'⎕WC'Edit'('Posn' 10 0)('Size'0 250)
 'FS.BOK'⎕WC'Button' 'OK'('Posn' 50 50)('Size'0 100)('Event' 'Select' 1)
 'FS.BAN'⎕WC'Button' 'Annuler'('Posn' 50 150)('Size'0 100)('Event'
'Select' 1)
aff:res←⎕DQ'FS'
 →(0=⍴res)/0
 →('FS.BAN'1⊃res)/0
 ⍝ Vérifier qu'un nom ait bien été saisi
 :If 0=⍴,FS.E1.Value
     'msg'⎕WC'MsgBox' 'GestPerso' 'Vous devez saisir un nom de société'
     'msg'⎕WS('Style' 'Warn')('Btns' 'OK')
     ⎕DQ'msg'
     →aff    ⍝ Réactiver l'écran
 :EndIf

 Numf←(chemin,'PersoGroup')⎕FSTIE 0
 ListSoc←⎕FREAD Numf 1
 ⍝ chercher si il y a une place vide à occuper
 place←(⍴¨,¨ListSoc)⍳0
 :If place>⍴ListSoc
     ListSoc,←⊂FS.E1.Text
     (4 1⍴'' '' 0 0)⎕FAPPEND Numf
 :Else
     ListSoc[place]←⊂FS.E1.Text
     (4 1⍴'' '' 0 0)⎕FREPLACE Numf(place+1)
 :EndIf
⍝ enregistrer le sommaire
 ListSoc ⎕FREPLACE Numf 1
 ⎕FUNTIE Numf

 GestPersoLoad place
```

- Supprimer la Société actuelle
Si cette option est choisie, on commence par demander confirmation de la société en reprécisant bien son nom dans la question.
Si la suppression est confirmée, on opère les changements suivants sur le fichier :
- dans la première composante on met iota zéro (⍳0) à la place du nom de la société effacée.
- on remplace la matrice de la société supprimée par une matrice de 4 lignes, une colonne, constituée de 2 chaînes vides et 2 zéros.
Enfin, on affiche la première société à l'écran.

Nouvelle ligne de menu dans GestPerso :

```
 'F1.MB.M1.DS'⎕WC'MenuItem' '&Supprimer la Société actuelle'('Event'
'Select' 'GestPersoDel')
```

Texte de la fonction GestPersoDel :

```
GestPersoDel;Numf;FS;ListSoc;res;msg;place
 ⍝ suppression d'une société
 'msg'⎕WC'MsgBox' 'GestPerso'('Confirmez-vous la suppression de la
société ',soc,' ?')
 'msg'⎕WS('Style' 'Query')('Btns' 'YES' 'NO')
 'msg'⎕WS('Event' 'MsgBtn1' 1)('Event' 'MsgBtn2' 1)
 res←⎕DQ'msg'
 →('MsgBtn2'2⊃res)/0

⍝ Charger la liste des sociétés
 Numf←(chemin,'PersoGroup')⊃FSTIE 0
 ListSoc←⎕frEAD Numf 1
 ⍝ supprimer la société de la liste
 place←ListSoc⍳⊂soc
 ListSoc[place]←⊂⍳0
 ListSoc ⎕frEPLACE Numf 1
 (4 1⍴'' '' 0 0)⎕frEPLACE Numf(place+1)
 ⎕fuNTIE Numf
⍝ Chercher la première société valide
 place←(0≠,⍴⍴¨ListSoc)⍳1
 GestPersoLoad place
```

Etant donné que la liste des sociétés peut désormais avoir des emplacements vides, il faut modifier la fonction GestPersoOpen de sorte qu'elle n'affiche que les sociétés valides :

```
 GestPersoOpen;Numf;FS;ListSoc;res;ls2
⍝ Ouvrir une des sociétés de la base
 Numf←(chemin,'PersoGroup')  ⎕FTIE 0
 ls2←ListSoc←⎕frEAD Numf 1
 ls2←(0≠,⍴⍴¨ls2)/ls2
 'FS'⎕WC'Form' 'Ouvrir une société ...'('Coord' 'Pixel')('Size' 200
240)('OkButton' 1)
 'FS.C1'⎕WC'Combo'('Items'ls2)('event' 'select' 1)
 FS.C1.SelItems←ls2¨⊂soc
 ⎕fuNTIE Numf
 res←⎕DQ'FS'
 →(0=⍴res)/0
 →('FS.C1'≠1⊃res)/0
 GestPersoLoad ListSoc⍳⊂FS.C1.Text
```

- Sauver et Quitter : Comme le bouton mais on sauve dans le fichier

Nouvelle ligne de menu dans GestPerso :

```
'F1.MB.M1.SQ'⎕WC'MenuItem' 'S&auver et Quitter'('Event' 'Select' 1)
```

Mise à jour du code de GestPerso concernant la sauvegarde :

```
∇ Sauver et Quitter
 Perso←F1.G1.Values
 tmp←(chemin,'PersoGroup')⊃FSTIE 0
 ListSoc←⎕fread tmp 1
 place←ListSoc⍳⊂soc
 Perso ⎕FREPLACE tmp(place+1)
 ⎕FUNTIE tmp
```

- Quitter sans sauver : Comme le bouton.

Nouvelle ligne de menu dans GestPerso :

```
'F1.MB.M1.QSS'⎕WC'MenuItem' '&Quitter sans sauver'('Event' 'Select' 1)
```

Mise à jour du code de GestPerso concernant la sortie sans sauvegarde :

```
∇ Quitter sans Sauver
→(('F1.BAQ'1⊃tmp)∨'F1.MB.M1.QSS'1⊃tmp)/0
```

5. Le département comptabilité voudrait un fichier texte des noms et salaires pour chaque société. Fichier sur 2 colonnes : nom;salaire

Ajoutez un menu "Export Salaires" juste avant "sauver et quitter" qui, au moyen d'un objet FileBox, permet de choisir précisément le nom et l'emplacement du fichier. Le nom proposé par défaut sera "Salaires société.txt"

Nouvelle ligne de menu dans GestPerso :

```
'F1.MB.M1.EX'⎕WC'MenuItem' 'E&xport salaires'('Event' 'Select'
'GestPersoExport')
```

Texte de la fonction GestPersoExport :

```
GestPersoExport d;tmp;numf;FB
 ⍝ Exporter au format texte
 'FB'⎕WC'FileBox' 'Exporter ...'chemin
 FB.File←'Salaires ',soc,'.txt'
 FB.FileMode←'Write'
 FB.Filters←⊂'*.txt' 'Fichier texte'
 'FB'⎕WS('Event' 'FileBoxOK' 1)('Event' 'FileBoxCancel' 1)
 tmp←⎕DQ'FB'
 ⍝ Sortir si "annuler"
 →('FileBoxCancel'2⊃tmp)/0

 ⍝ Effacer le fichier si il existe
 ⎕Trap 0
   numf←(FB.Directory,FB.File)⎕NTIE 0
   (FB.Directory,FB.File)⎕NERASE numf
 :EndTrap

 ⍝ Créer fichier
 numf←(FB.Directory,FB.File)⎕NCREATE 0
 tmp←⍉F1.G1.Values[1 4;]     ⍝ Récupérer valeurs et transposer
 tmp[;1]←tmp[;1],¨';'        ⍝ Ajouter séparateur
 tmp←((↑tmp),⎕AV[4]),⎕AV[3] ⍝ Convertir en texte et ajouter sauts de
lignes
 (,tmp)⎕NAPPEND numf         ⍝ Linéariser et enregistrer
 ⎕NUNTIE numf                ⍝ Libérer fichier
 'FB'⎕WC'msgbox' 'Exporter'('Fichier ',FB.Directory,FB.File,' exporté')
 FB.Style←'Info'
 FB.Btns←'OK'
 ⎕DQ'FB'
```

Texte final de la fonction GestPerso :

```
GestPerso;F1;tmp;PersoLoc;soc;chemin;ListSoc;place
⍝ Gestion du personnel
 'F1'⎕WC'Form' 'Personnel MyStyle Ltd'('Size' 400 600)('Coord' 'Pixel')
 ⍝ Chemin du ws
 chemin←(⊃,/¯1↓(1,1↓⎕WSID='\')⊂⎕WSID),'\'
 'F1.G1'⎕WC'Grid'('Posn' 0 0)('Size' 150 600)('Values'Perso)
 F1.G1.RowTitles←'Nom' 'Sexe' 'Age' 'Sal. annuel'
 'F1.G1.E1'⎕WC'Edit'('FieldType' 'Char')
 'F1.G1.C1'⎕WC'Combo'('Style' 'Drop')('Items'(2 1⍴'HF'))
 'F1.G1.E2'⎕WC'Edit'('FieldType' 'Numeric')
 F1.G1.CellTypes←(⍴F1.G1.Values)⍴3
 F1.G1.CellTypes[1;]←1
 F1.G1.CellTypes[2;]←2
 F1.G1.Input←'F1.G1.E1' 'F1.G1.C1' 'F1.G1.E2'
 F1.G1.AutoExpand←0 1

 'F1.M1'⎕WC'Menu'
 'F1.M1.Del'⎕WC'MenuItem' 'Supprimer'('Event' 'Select' 1)
 'F1.M1.Sel'⎕WC'MenuItem' 'Sélectionner'('Event' 'Select' 0)
 'F1.G1'⎕WS('Event' 'CellDown' 'GestPerso_sel')

 'F1.LS'⎕WC'Label' 'Statistiques'('Posn' 170 0)
 'F1.G2'⎕WC'Grid'('Posn' 200 0)('Size' 100 600)('Values'(3 3⍴0))
 F1.G2.RowTitles←'Effectif' 'Masse salariale' 'Sal. moyen'
 F1.G2.ColTitles←'Hommes' 'Femmes' 'Total'
 'F1.G1'⎕WS('Event' 'CellChanged' 'GestPerso_Stats')
 GestPerso_Stats 1

 'F1.LS2'⎕WC'Label' 'Pourcentage de Variation :'('Posn' 310 10)('Size'⍬
150)
 'F1.E1'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 310 170)('Size'⍬ 100)
 'F1.LS3'⎕WC'Label' 'Montant maxi d''augmentation :'('Posn' 340
10)('Size'⍬ 150)
 'F1.E2'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 340 170)('Size'⍬ 100)
 'F1.BC'⎕WC'Button' 'Lancer simulation'('Posn' 310 300)('Size'⍬
100)('Event' 'Select' 'GestPerso_simu')
 'F1.BR'⎕WC'Button' 'Fermer simulation'('Posn' 340 300)('Size'⍬
100)('Event' 'Select' 'GestPerso_simu')

 PersoLoc←Perso

 'F1.BSQ'⎕WC'Button' 'Sauver et Quitter'('Posn' 310 450)('Size'⍬
100)('Event' 'Select' 1)
 'F1.BAQ'⎕WC'Button' 'Quitter sans Sauver'('Posn' 340 450)('Size'⍬
100)('Event' 'Select' 1)

 'F1.MB'⎕WC'MenuBar'
 'F1.MB.M1'⎕WC'Menu' '&Fichier'
 'F1.MB.M1.OS'⎕WC'MenuItem' '&Ouvrir Société'('Event' 'Select'
'GestPersoOpen')
 'F1.MB.M1.NS'⎕WC'MenuItem' '&Nouvelle Société'('Event' 'Select'
'GestPersoNew')
 'F1.MB.M1.DS'⎕WC'MenuItem' '&Supprimer la Société actuelle'('Event'
'Select' 'GestPersoDel')
 'F1.MB.M1.EX'⎕WC'MenuItem' 'E&xport salaires'('Event' 'Select'
'GestPersoExport')
 'F1.MB.M1.SQ'⎕WC'MenuItem' 'S&auver et Quitter'('Event' 'Select' 1)
 'F1.MB.M1.QSS'⎕WC'MenuItem' '&Quitter sans sauver'('Event' 'Select' 1)

 GestPersoLoad 1

 tmp←⎕DQ'F1'
 →(0=⍴tmp)/0
```

```
⍝ Quitter sans Sauver
→(('F1.BAQ'1⊃tmp)∨'F1.MB.M1.QSS'1⊃tmp)/0

⍝ Sauver et Quitter
Perso←F1.G1.Values
tmp←(chemin,'PersoGroup')⎕FSTIE 0
ListSoc←⎕FREAD tmp 1
place←ListSoc⍳⊂soc
Perso ⎕FREPLACE tmp(place+1)
⎕FUNTIE tmp
```