PSION

Bulletin Technique

Octobre Novembre Decembre 1989 La revue du support technique d'Aware

ORGANISEUR II

- La gestion des enregistrements
- le mode Calc par l'OPL
- Des chiffres en lettres
- La gestion des dates
- Un grapheur
- Spreadsheet et Multiplan

PC4

- Restructurer un fichier Archive
- Un outils très puissant : Easel

Versions en cours

Edité par : Aware
7/9, rue des Petites Ecuries
75010 PARIS
Tél. : (1) 45.23.21.12
Télex : 281 941
Fax : (1) 45.23.02.37

3616 code AWARE

Aware ne disposait jusqu'à aujourd'hui que du Bulletin Technique pour diffuser des exemples d'application de l'OPL et répondre à une demande toujours très forte en matière de programmation. Pour répondre à cet état de fait, nous avons décidé de créer le service Minitel 36 16 code Aware dont une des ambitions est de propager l'OPL (et tout le savoir-faire s'y rapportant) auprès des utilisateurs de l'Organiseur II.

La mission première du 36 16 Aware est l'assistance technique, autrement dit, ce service a pour but la résolution de vos problèmes techniques et l'échanges d'idées entre utilisateurs.

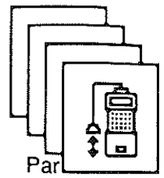
Si vous posez vos questions dans l'option Support Technique, nos techniciens vous fourniront dans les délais les plus brefs des réponses précises et fiables.
Si par contre vous laissez un message dans l'option Annonce, ce sont les autres utilisateurs qui seront susceptibles de vous répondre.

Enfin, si seule la curiosité vous guide, vous pouvez toujours consulter soit les "Questions d'Intérêt général" dans le Support Technique soit les Annonces déjà existantes ; les problèmes rencontrés par nos utilisateurs sont souvent passionnants et nous espérons bien que chaque jour le 36 16 Aware vous offrira un panorama renouvelé d'astuces et de solutions !

En parlant d'astuces et de solutions, je vous laisse maintenant prendre connaissance des articles de ce présent numéro et vous souhaite des heures riches d'enseignements.

Philippe Delaplace

LES COMMANDES DE GESTION D'ENREGISTREMENTS



Par
Gilles JEAN

Nous avons vu, dans le précédent Bulletin Technique, les commandes pouvant servir dans l'élaboration d'une base de données sur votre Organiseur. L'objet de cet article est de mettre maintenant à votre disposition les commandes qui permettent la manipulation des enregistrements de/des fichier(s).

Voici la liste des commandes que nous allons utiliser dans les exemples qui suivent :

```
FIRST
Syntaxe : first
Exemple : first
```

Cette commande permet de se positionner sur le premier enregistrement du fichier activé par les commandes CREATE, OPEN, USE. Si le fichier ne contient pas d'enregistrement, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.

```
NEXT
Syntaxe : next
Exemple : next
```

Cette commande permet de se positionner sur l'enregistrement suivant (de celui en mémoire) du fichier activé par les commandes CREATE, OPEN, USE. Si le fichier ne contient pas d'enregistrement ou si l'on se trouve sur la fin du fichier, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.

```
BACK
Syntaxe : back
Exemple : back
```

Cette commande permet de se positionner sur l'enregistrement précédent (de celui en mémoire) du fichier activé par les commandes CREATE, OPEN, USE. Si le fichier ne contient pas d'enregistrement ou si l'on se trouve sur la fin du fichier, aucune erreur ne sera générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.

```
LAST
Syntaxe : last
Exemple : last
```

Cette commande permet de se positionner sur le dernier enregistrement du fichier activé par les commandes CREATE, OPEN, ou USE. Si le fichier ne contient pas d'enregistrement, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.


```
FIND
Syntaxe : f%=find(exp$)
Exemple : f%=find("75010")
```

Cette commande permet d'effectuer une recherche, à partir de l'enregistrement en cours jusqu'à la fin du fichier, de la chaîne spécifiée entre parenthèses. Si cette commande trouve la chaîne correspondante dans un des enregistrements, f% correspond alors au numéro de position de l'enregistrement et le place en vigueur. Si le fichier ne contient pas d'enregistrement ou si la fin du fichier est détectée, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.

```
FINDW
Syntaxe : f%=findw(exp$)
Exemple : f%=findw("75*PARIS")
```

Cette commande permet d'effectuer une recherche, à partir de l'enregistrement en cours jusqu'à la fin du fichier, de la chaîne spécifiée entre parenthèses. Si cette commande trouve la chaîne correspondante dans un des enregistrements, f% correspond alors au numéro de position de l'enregistrement et le place en vigueur. Si le fichier ne contient pas d'enregistrement ou si la fin du fichier est détectée, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie. Il est à préciser que cette commande n'est exécutable que sur les modèles LZ et LZ64. L'utilisation des jokers * et + permet une recherche multi-critères. Dans notre exemple, nous recherchons à la fois sur la ville PARIS et sur le code postal 75.

```
POSITION
Syntaxe : position exp%
Exemple : position 5
```

Cette commande fait de l'enregistrement numéro exp%, l'enregistrement en vigueur dans le fichier activé par les commandes CREATE,OPEN ou USE. Si le numéro spécifié est supérieur au nombre total d'enregistrements dans le fichier en vigueur, le dernier enregistrement devient alors celui en vigueur. Si le fichier ne contient pas d'enregistrement, aucune erreur n'est générée mais l'enregistrement en vigueur est vide et la fonction EOF est vraie.

```
POS
Syntaxe : p%=pos
Exemple : p%=pos
```

Cette fonction calcule le numéro de l'enregistrement en vigueur dans le fichier activé par les commandes CREATE, OPEN ou USE.

```
EOF
Syntaxe : eof
Exemple : if eof :cls :print"*Fin du fichier*" :get
```

Cette fonction active l'indicateur (pointeur) spécifiant que la fin du fichier est atteinte lorsque le programme essaie de dépasser le dernier enregistrement du fichier activé par les commandes CREATE, OPEN ou USE.

```
DISP
Syntaxe : d%=disp(exp%,exp$)
Exemple : d%=disp(-1,"")
```

Cette commande permet d'afficher à l'écran l'enregistrement en vigueur pour le fichier activé par les commandes CREATE, OPEN ou USE. Comme cela est le cas avec le calepin électronique, vous pouvez examiner le contenu à l'aide des touches du curseur. Cette commande a d'autres utilisations que vous trouverez documentées dans votre manuel de programmation.

```
APPEND
Syntaxe : append
Exemple : input a.a$ :append
```

Cette commande permet d'insérer un enregistrement dans le fichier activé par les commandes CREATE, OPEN ou USE. Cet enregistrement est alors enregistrement en vigueur.

```
UPDATE
Syntaxe : update
Exemple : edit a.a$ :update
```

Cette commande permet de modifier l'enregistrement en vigueur dans le fichier activé par les commandes CREATE, OPEN ou USE. Cet enregistrement est alors positionné en fin de fichier et déclaré enregistrement en vigueur.

```
ERASE
Syntaxe : erase
Exemple : erase
```

Cette commande permet de supprimer l'enregistrement en vigueur dans le fichier activé par les commandes CREATE, OPEN ou USE. Si l'enregistrement détruit était le dernier du fichier, l'enregistrement en vigueur est vide et la fonction EOF (fin du fichier) est vraie.

```
TRAP
Syntaxe : trap <commande>
Exemple : trap open"b:articles",a,code$,desc$,pa,pv,qte
```

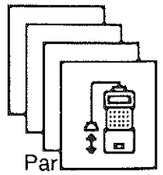
Cette commande peut précéder toutes les commandes de gestion de fichiers. Les erreurs alors provoquées par la commande associée sont interceptées, pour permettre l'exécution de la ligne suivante du programme.

L'exemple ci-dessous est un applicatif très simple illustrant la manipulation des enregistrements d'un fichier. Vous pouvez insérer (Ins), modifier (Mod), détruire (Det), trouver (Tro), suivant (Sui), précédent (Pre), premier (Fir), dernier (Las) et visualiser (Vis) l'enregistrement en vigueur.

```
test5:
local v%,t$(32)
open"a:main",a,a$,b$
do
cls
at 1,2
if eof
beep 99,99
print"*Fin du fichier*"
else
print left$(a.a$+" "+a.b$,16)
endif
v%=view(1,"Ins,Tro,Sui,Pre,Mod,Det,Fir,Las,Vis")
if v%=1
return
elseif v%=%I
cls
print"Rub1:",
input a.a$
print"Rub2:",
input a.b$
append
elseif v%=%T
cls
print"Trouver:"
trap input t$
if err=0
first
find(t$)
endif
elseif v%=%S
next
elseif v%=%P
back
elseif v%=%M and not(eof)
cls
print"Rub1:",
edit a.a$
print"Rub2:",
edit a.b$
update
elseif v%=%D and not(eof)
erase
elseif v%=%F
first
elseif v%=%L
last
elseif v%=%V and not(eof)
disp(-1,"")
endif
until 0
```

Note : la boucle DO ... UNTIL 0 permet de répéter la boucle indéfiniment. Il est donc important de gérer vous même les sorties de programmes.

L'OPTION CALC PAR L'OPL



Par
Gilles JEAN

Il y a déjà bien longtemps, nos techniciens avaient mis au point une possibilité d'appeler une option du menu principal (voir bulletin technique 38-39-40 des mois d'Octobre, Novembre et Décembre 1987). Cet article avait à l'époque (et même aujourd'hui) soulevé bien des controverses dans la mesure où la fonction CALC (calculatrice) ne pouvait être invoquée de cette sorte. Le programme listé ci-dessous va nous permettre de résoudre ce problème. Le principe en est relativement simple. La valeur de la variable CALCADD% est l'adresse dans l'Organiseur II de la routine de la fonction CALC. Celle-ci est obtenue en regardant à l'adresse $2002 qui contient le début du menu de l'Organiseur II. Sur le modèle LZ64, le premier octet représente la longueur de la première option du menu (option RECH). Les quatre octets suivants représentent le littéral de l'option R-E-C-H et les deux suivants l'adresse de la routine RECH. La procédure FINDTOP% permet de retourner la valeur à la variable CALCADD%.

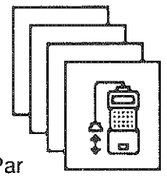
```
calc:
local a%(55),calcadd%
rem ***********************************************
rem ***** Merci à l'équipe PSION pour l'aide  *****
rem **** apporté dans la réalisation de cette *****
rem *****           routine                   *****
rem ***********************************************
a%(1)=$0118
a%(2)=$dca5
a%(3)=$b320
a%(4)=$4018
a%(5)=$8c03
a%(6)=$e82b
a%(7)=$1cfe
a%(8)=$200c
a%(9)=$bc20
a%(10)=$0e26
a%(11)=$14bc
a%(12)=$2010
a%(13)=$260f
a%(14)=$bc20
a%(15)=$1226
a%(16)=$0abc
a%(17)=$2014
a%(18)=$2605
a%(19)=$bc20
a%(20)=$1627
a%(21)=$0139
a%(22)=$dea9
a%(23)=$3cfe
a%(24)=$2065
a%(25)=$3cde
a%(26)=$a53c
a%(27)=$ff20
a%(28)=$65de
a%(29)=$a73c
a%(30)=$deb6
a%(31)=$3cfe
a%(32)=$23e2
a%(33)=$3cde
a%(34)=$b03c
a%(35)=$deb2
a%(36)=$3cde
a%(37)=$b43c
a%(38)=$18ad
a%(39)=$0038
a%(40)=$deb4
a%(41)=$38df
a%(42)=$b238
a%(43)=$dfb0
a%(44)=$38ff
a%(45)=$23e2
a%(46)=$38df
a%(47)=$b638
a%(48)=$dfa7
a%(49)=$38df
a%(50)=$a538
a%(51)=$ff20
a%(52)=$6538
a%(53)=$dfa9
a%(54)=$3f46
a%(55)=$3900
calcadd%=findtop%:
usr(addr(a%()),calcadd%)


findtop%:
local calcadd%,i%,a%,n%,s$(5),o$(4)
i%=peekw($2002)
do
a%=peekb(i%)
if a%=0
return 0
endif
if a%<>4
i%=i%+a%+3
continue
endif
s$=""
n%=1
do
s$=s$+chr$(peekb(i%+n%))
n%=n%+1
until n%=5
rem test de la langue en cours - ici FR et GB
if peekb($2186)<2
o$="CALC"
rem test de la langue en cours - ici RFA
elseif peekb($2186)=2
o$="RECH"
rem test de la langue en cours - ni FR ni GB ni RFA
else stop
endif
if upper$(s$)<>o$
i%=i%+a%+3
continue
endif
calcadd%=i%+a%+1
calcadd%=peekw(calcadd%)
return(calcadd%)
until 0
```

Il faut toutefois noter que ces routines permettent de lancer les autres applications système de votre Organiseur II. Les seules modifications à faire consistent à changer :
- La valeur du test sur la longueur de la chaîne recherchée dans le menu a% (ligne 9).
- La valeur de la variable n$ (ligne 18)
- le test de la construction de s$ en boucle (ligne 21)

ATTENTION
L'absence de l'option CALC du menu principal entraine une erreur système (trap). Il est important de spécifier que ces deux routines ont été testées sur CM, XP, LZ et LZ64 en Anglais, Français et Allemand. Si votre Organiseur II peut être configuré dans une autre langue, cette routine ne fonctionnera pas.

REPRESENTATION ALPHANUMERIQUE DE CHIFFRE



Par
Frédéric MARCHESI

Un petit programme qui va vous permettre de convertir un nombre provenant d'une rubrique numérique en son équivalent littéral. Ce module peut être exploité pour l'édition de lettre-chèque par exemple, lorsque le montant doit impérativement figurer en toutes lettres. Ce programme est composé de trois procédures NUMALPHA, CENTDIX, UNITE. NUMALPHA permet de convertir les parties million et millier de la somme. CENTDIX convertit les parties centaine et dizaine de la somme. UNITE traite quant à lui les unités de la somme entre 1 et 19. Vous noterez qu'il est possible d'entrer deux décimales au chiffre et ainsi les traiter comme la partie centime du montant.

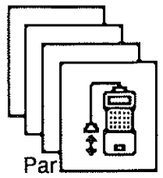
```
numalpha:
global mont$(255),va,d11,d12,d%
do
cls
print"Valeur Num ?"
trap input va
if err
return
else
mont$=""
if intf(va)=0
mont$="zero "
endif
va=intf(va*100)/100
d11=intf(va/1000000)
if d11>0
centdix:
mont$=mont$+"million"
if d11>1
mont$=mont$+"s"
endif
mont$=mont$+" "
if intf(va)/1000000=intf(va/1000000)
mont$=mont$+"de "
endif
endif
d11=intf(va/1000)-(d11*1000)
if d11>0
if d11>1
centdix:
endif
mont$=mont$+"mille "
endif
d11=intf(va)-(intf(va/1000)*1000)
if d11>0
centdix:
endif
mont$=mid$(mont$,1,len(mont$)-1)
if (d11/100=intf(d11/100)) and (intf(va)>100)
mont$=mont$+"s"
elseif (d11-(intf(d11/100)*100)=80)
mont$=mont$+"s"
endif
mont$=mont$+" franc"
if intf(va)>1
mont$=mont$+"s "
else
mont$=mont$+" "
endif
d11=(va-intf(va))*100
if d11>0
mont$=mont$+"et "
centdix:
mont$=mont$+"centime"
if d11>1
mont$=mont$+"s"
endif
endif
endif
do
cls
at 1,2
beep 99,99
print va
d%=view(1,mont$)
until d%=13
until 0


centdix:
d12=intf(d11/100)
if d12>0
if d12>1
unite:
endif
mont$=mont$+"cent "
endif
d12=intf(((d11/100)-intf(d11/100))*10)
if d12=2
mont$=mont$+"vingt "
elseif d12=3
mont$=mont$+"trente "
elseif d12=4
mont$=mont$+"quarante "
elseif d12=5
mont$=mont$+"cinquante "
elseif d12=6 or d12=7
```

```
mont$=mont$+"soixante "
elseif d12=8 or d12=9
mont$=mont$+"quatre-vingt "
endif
if (d12>1) and (d12<9) and (d11-(10*int(d11/10))=1)
mont$=mont$+"et "
endif
if (d12=1) or (d12=7) or d12=9
d12=10
else
d12=0
endif
d12=d11-(intf(d11/10)*10)+d12
unite:


unite:
if d12=1
mont$=mont$+"un "
elseif d12=2
mont$=mont$+"deux "
elseif d12=3
mont$=mont$+"trois "
elseif d12=4
mont$=mont$+"quatre "
elseif d12=5
mont$=mont$+"cinq "
elseif d12=6
mont$=mont$+"six "
elseif d12=7
mont$=mont$+"sept "
elseif d12=8
mont$=mont$+"huit "
elseif d12=9
mont$=mont$+"neuf "
elseif d12=10
mont$=mont$+"dix "
elseif d12=11
mont$=mont$+"onze "
elseif d12=12
mont$=mont$+"douze "
elseif d12=13
mont$=mont$+"treize "
elseif d12=14
mont$=mont$+"quatorze "
elseif d12=15
mont$=mont$+"quinze "
elseif d12=16
mont$=mont$+"seize "
elseif d12=17
mont$=mont$+"dix sept "
elseif d12=18
mont$=mont$+"dix huit "
elseif d12=19
mont$=mont$+"dix neuf "
endif
```

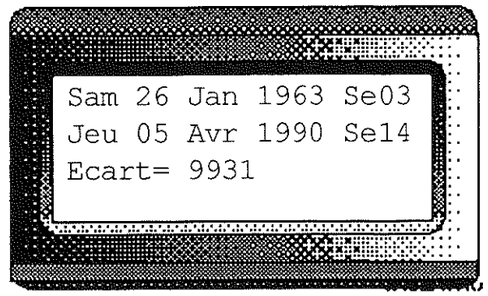
Et maintenant, à vous de jouer...



Par
Gilles JEAN

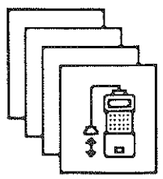
LES FONCTIONS DE GESTION DE DATE DU LZ

Le langage de programmation de l'Organiseur II modèle LZ met maintenant à notre disposition des fonctions permettant d'effectuer de nombreux calculs sur les dates. Voici un programme très amusant qui vous permettra d'obtenir rapidement l'écart, exprimé en jours, entre deux dates. Le réglage des valeurs des dates se fait avec les touches flèches du clavier de votre Organiseur.




```
xdate:
local g%,a%,b%,p%,f%,i%,j1,j2,mo1,mo2,a1,a2,e1,e2
p%=1 :j1=day :mo1=month :a1=year :j2=day :mo2=month :a2=year
onerr fin::
f::
do
if b%=0 or b%=1 and i%=0
at 1,1
print dayname$(dow(j1,mo1,a1)),mid$(gen$(100+j1,3),2,2),month$(mo1)
,a1,"Se";mid$(gen$(100+week(j1,mo1,a1),3),2,2)
endif
if b%=0 or b%=2 and i%=0
at 1,2
print dayname$(dow(j2,mo2,a2)),mid$(gen$(100+j2,3),2,2),month$(mo2)
,a2,"Se";mid$(gen$(100+week(j2,mo2,a2),3),2,2)
endif
e1=days(j1,mo1,a1) :e2=days(j2,mo2,a2)
if i%=0
at 1,3
print"Ecart=",abs(e1-e2);"      "
endif
i%=1
if p%<1
p%=6
elseif p%>6
p%=1
endif
if p%=1 or p%=4
a%=5
elseif p%=2 or p%=5
a%=8
elseif p%=3 or p%=6
a%=12
endif
if p%<4
```

```
b%=1
elseif p%>3
b%=2
endif
at a%,b%
kstat 1
cursor on
g%=get
if g%=1
return
elseif g%=3
f%=1 :i%=0
elseif g%=4
f%=-1
i%=0
elseif g%=5
p%=p%-1
elseif g%=6
p%=p%+1
endif
if f%=1 or f%=-1
if p%=1
j1=j1+f%
elseif p%=2
mo1=mo1+f%
elseif p%=3
a1=a1+f%
elseif p%=4
j2=j2+f%
elseif p%=5
mo2=mo2+f%
elseif p%=6
a2=a2+f%
endif
f%=0
endif
if mo1<1
mo1=12
elseif mo1>12
mo1=1
elseif mo2<1
mo2=12
elseif mo2>12
mo2=1
endif
if a1<1900
a1=2155
elseif a1>2155
a1=1900
elseif a2<1900
a2=2155
elseif a2>2155
a2=1900
endif
until 0
fin::
if err=247
if p%<4
j1=1
elseif p%>3
j2=1
endif
i%=0
else ert:
endif :goto f::
```

Par Philippe DELAPLACE

GRAPHEUR SUR ORGANISEUR 4 LIGNES

Le programme suivant permet d'afficher sur l'écran d'un LZ ou d'un LZ 64 un histogramme d'un maximum de 20 colonnes, la hauteur de chaque colonne pouvant varier entre 32 valeurs différentes. Cette procédure vous sera particulièrement utile si vous possèdez déjà un fichier de données numériques que vous voulez visualiser.
Pour permettre une adaptation optimale avec vos fichiers ou vos procédures déjà existantes, le grapheur est divisé en une procédure appelante VUBOOT et trois sous procédures UDGVU, VUCALC et TRAME qui doivent être appelées par VUBOOT.

```
UDGVU:
udg 0,21,11,21,11,21,11,21,11
udg 1,0,0,0,0,0,0,0,31
udg 2,0,0,0,0,0,0,31,21
udg 3,0,0,0,0,0,31,21,11
udg 4,0,0,0,0,0,31,21,11,21
udg 5,0,0,0,0,31,21,11,21,11
udg 6,0,0,0,31,21,11,21,11,21
udg 7,0,31,21,11,21,11,21,11
return
```

UDGVU définit 8 caractères qui vont être utilisés par VUCALC et TRAME pour l'affichage de l'histogramme.

```
TRAME:
at 1,1 :print rept$(chr$(1),20)
at 1,2 :print rept$(chr$(1),20)
at 1,3 :print rept$(chr$(1),20)
at 1,4 :print rept$(chr$(1),20)
return
```

TRAME affiche à l'écran 4 lignes horizontales qui serviront de repères (et d'agrément esthétique) pour l'histogramme.

```
VUCALC:
local r%,n%,h%
h%=int(32*((x-x0)/(x4-x0)))
if h%>0
n%=h%/8
r%=h%-8*n%
endif
if n%>0
at l%,4 :print chr$(0)
endif
if n%>1
at l%,3 :print chr$(0)
endif
if n%>2
at l%,2 :print chr$(0)
endif
if n%>3
if r%=0
at l%,1 :print chr$(0)
else
at l%,1 :print chr$(245)
```

```
endif
endif
if h%>0 and h%<32 and r%>0
at l%,4-n% :print chr$(r%)
elseif h%>39
beep 40,154
elseif h%<-7
beep 40,850
endif
if h%<0
at l%,4 :print chr$(170)
endif
return
```

La procédure VUCALC constitue le coeur du grapheur . Son fonctionnement en est le suivant : la valeur x (valeur à visualiser) est comparée aux valeurs x0 et x4 (respectivement les valeurs minimales et maximales affichables à l'écran). h% est le résultat de cette opération et correspond au nombre d'échelons empilés dont sera constitué un "bâton" de l'histogramme . n% et r% sont le dividende et le reste de la division de h% par 8 (par exemple h%=21 donne n%=2 et r%=5). VUCALC empile ensuite par le bas n% caractères chr$(0) représentant 8 échelons empilés et le caractère chr$(r%) constitué de r% échelons empilés. La variable l% détermine la position à laquelle est affiché le "bâton". En outre si la valeur x ne rentre pas dans l'intervalle [x0,x4] d'affichabilité, VUCALC le signale à l'affichage et (en cas de franc dépassement) par un signal sonore.

```
VUBOOT:
global x,x0,x4,l%
open "a:nom",a,num,chaine$
rem ** à la place de nom, écrivez **
rem **  le nom de votre fichier   **
udgvu:
top::
l%=0
trame:
x0=0
x4=1000
rem ** x0 et x4 doivent encadrer **
rem ** vos valeurs x à afficher  **
l%=l%+1
do
x=a.num
vucalc:
beep 1,1000
next
until l%=20 or eof
beep 40,386
if eof
get
at 1,1 :print "Fin de Fichier"
first
endif
if get<>1
goto top::
endif
at 12,4 :print "au revoir"
pause 20
```


Cette procédure VUBOOT ne doit pas être transcrite littéralement ; c'est à vous de savoir quelle zone de quel fichier vous voulez utiliser et adapter VUBOOT en conséquence. Le programme tel qu'il est écrit accède à la première zone (qui est de type numérique) du fichier "a:nom".
Si les données auxquelles vous voulez accéder sont enregistrées sur la deuxième zone et sous forme de chaînes, il vous faudra substituer à la ligne

```
x=a.num
```

la ligne

```
x=val(a.chaîne$)
```

Autre point à soulever : Si vous utiliser les 20 colonnes de l'écran de l'Organiseur, il ne reste pas d'espace vacant pour afficher une éventuelle légende ; seul remède : réduire le nombre de "bâtons" affichés par le grapheur en substituant à la ligne

```
until l%=20 or eof
```

par exemple, la ligne

```
until l%=12 or eof
```

Vous obtiendrez alors un affichage 12 colonnes.

PS : Les données suivantes peuvent vous être utiles :
Un échelon supplémentaire représente une augmentation de x de :
(x4-x0)/32
Les lignes de trames indiquent des valeurs (respectivement dehaut en bas) de :
(x0+31*x4)/32
(9*x0+23*x4)/32
(23*x0+9*x4)/32
(31*x0+x4)/32

SPREADSHEET ET MULTIPLAN

Nous allons dans cet article vous donner toutes les clefs pour réaliser des transferts de tableaux Spreadsheet vers multiplan et vice versa. Il va s'en dire que nous tranferrons également les formules. Seule limite : la taille des tableaux, puisque nous avons la possibilité de réaliser dans Spreadsheet des tableaux d'une taille maximale de 26 colonnes sur 99 lignes.
Précisons que tous ces tests ont été réalisé avec la version 3.0 de Multiplan.
Configurons les paramètres de communication série. Activons notre interface série en appuyant deux fois sur la touche "ON" pour charger l'option Comms dans le menu principal, et selectionnons cette option.

Il vous faudra configurer les paramètres de transfert de votre liaison série comme suit :

Bauds	9600
Parite	sans
Bits	8

Stop	1
Hand	xon
Protocole	PSION
Echo	HOTE
Larg.	SANS
Timeout	SANS
Rfdl	<CR><LF>
Rfdf	<SUB>
Rtrn	SANS
Tfdl	<CR><LF>
Tfdf	<SUB>
Ttrn	SANS

Bien sûr cette configuration est à utiliser avec notre logiciel de communication Comms-Link pour votre Macintosh ou votre PC.

Export Spreadsheet :
Assurez vous que vous avez sauvegardé votre fichier de travail en fichier de stockage afin de donner un nom au fichier que vous désirez transférer.
Lancez le logiciel de communication Comms-Link sur votre PC, et utilisez ensuite les options "import" et "export" du menu "Fic" de Spreadsheet .
Attention cet ordre est important, si vous ne le respectez pas, le menu des formats de Spreadsheet n'apparaîtra pas et vous sortirez sur un message "erreur de connexion". On vous demande ensuite le type de format que vous désirez utiliser pour réaliser votre transfert, si vous choisissez l'option "WKS", vous pourrez exporter ou importer vos formules en plus de vos datas insérées dans vos cellules. Pour plus de détails sur les formats de fichier de Spreadsheet, vous pouvez consulter l'annexe C de votre manuel tableur à la page 141. Ensuite précisez le nom du fichier que vous désirez exporter sans l'extension. Une fois ce transfert réalisé, vous allez récupérer un fichier avec l'extension ".WKS", fichier que vous ouvrirez avec Multiplan.

Utilisation de Multiplan:
Sélectionnez les options suivantes des menu de Multiplan

```
Commande: Alpha Blanc Calcul Detruit Edite Format Guide Insère LIT-Ecrit Mouv Nom Options Protège Quitte Recopie Sortie Tri Utilise Vers Xtern ZoneFenêtre
```

Ensuite, vous vous retrouvez dans le menu Lit-Ecrit:

```
LitEcrit: Charge Sauvegarde Effacer-écran Détruit Options Renomme Transfert

LitEcrit Options mode: Normal Symbolique Autre
```

Après avoir sélectionné l'option Autre, Multiplan vous renvoie à son menu principal, voici les options à sélectionner.

```
Commande: Alpha Blanc Calcul Detruit Edite Format Guide Insère LIT-Ecrit Mouv Nom Options Protège Quitte Recopie Sortie Tri Utilise Vers Xtern ZoneFenêtre
```


puis,

```
LitEcrit: Charge Sauvegarde Effacer-écran Détruit Options Renomme Transfert
```

A ce stade-là, vous devez insérer le nom de votre fichier suivi de son extension. Ex: " test.Wks "

Vous avez à ce moment là, récupéré votre tableau Spreadsheet dans Multiplan avec toutes ses formules.

Import Spreadsheet :
Voyons maintenant la manipulation inverse, transfert multiplan vers Spreadsheet (les manipulations sont similaires à ce que nous avons vu précédemment). Sélectionnez les options suivantes

```
Commande: Alpha Blanc Calcul Detruit Edite Format Guide Insère LIT-Ecrit Mouv Nom Options Protège Quitte Recopie Sortie Tri Utilise Vers Xtern ZoneFenêtre
```

Ensuite, vous vous retrouvez dans le menu Lit-Ecrit:

```
LitEcrit: Charge Sauvegarde Effacer-écran Détruit Options Renomme Transfert

LitEcrit Options mode: Normal Symbolique Autre
```

Après avoir sélectionné l'option Autre, Multiplan vous renvoie à son menu principal, voici les options à sélectionner.

```
Commande: Alpha Blanc Calcul Detruit Edite Format Guide Insère LIT-Ecrit Mouv Nom Options Protège Quitte Recopie Sortie Tri Utilise Vers Xtern ZoneFenêtre
```

puis,

```
LitEcrit: Charge Sauvegarde Effacer-écran Détruit Options Renomme Transfert
```

A ce stade-là, vous devez insérer le nom de votre fichier suivi de son extension. Ex: " test.Wks".
Insérez le nom du fichier avec l'extenion ".WKS". Ex:" Test.Wks ".
Vous avez à ce moment-là, dans votre directory votre fichier avec l'extension ".WKS". Vous devez donc lancer le logiciel CL, et seulement ensuite, accéder aux options "Fic" et "Import" de Spreadsheet. Spécifiez le format ".WKS" et ensuite le nom de votre fichier Multiplan sans l'extension. Votre grille apparaît avec le nouveau fichier que vous venez d'importer.

Maintenant, à vos transferts.

RESTRUCTURATION D'UN FICHIER

Par
Frédéric MARCHESI

Dans un bulletin technique précédent,un article traitait de la restructuration de fichiers grâce au langage TSL. Dans cet article nous allons traiter la restructuration d'un fichier grâce au langage d'Archive.

Pour représenter notre problème il faut imaginer une société qui crée un fichier 'CLIENT' alors que les besoins ne sont pas encore définis, ce fichier contient les rubriques suivantes.

```
Fichier CLIENT

Numcl$          Num compte client
Societe$        Nom de la société
Nom$            Nom de contact
Prenom$         Prénom de ce dernier
Adresse$        Adresse de la société
Cp$             Code postal
Ville$          Ville
Tel$            Téléphone
Solde           Solde du client
```

Après un certain temps on s'aperçoit qu'il manque en fait plusieurs rubriques à notre fichier. Il nous faudrait par exemple une rubrique commentaires, ainsi qu'une deuxième ligne d'adresse.

Archive ne donne pas directement la possibilité d'ajouter des rubriques ou d'en modifier la structure, donc nous allons devoir écrire une petite procédure qui va traiter notre fichier pour lui donner le format que l'on désire sans perdre les données déjà insérées.
En quelques lignes relativement simples notre problème va être résolu, il nous faut tout d'abord un nouveau fichier client qui va contenir cette fois ci toutes les rubriques nécessaires et que nous allons nommer 'CLBIS'.

Le fichier CLBIS aura la structure suivante :

Fichier CLBIS

```
Numcl$          Num compte client
Societe$        Nom de la société
Nom$            Nom de contact
Prenom$         Prénom de ce dernier
Adress1$        Adresse de la société
Adress2$        Adresse de la société   (Nouvelle ligne d'adresse)
Cp$             Code postal
Ville$          Ville
Tel$            Téléphone
Solde           Solde du client
Comment$        Commentaire         (Nouvelle rub commentaires)
```

Voici la nouvelle structure de notre fichier 'CLIENT'. Maintenant que nous avons défini et créé une nouvelle structure, nous allons devoir ouvrir les deux fichiers en même temps pour passer les enregistrements d'un fichier à l'autre sans rien perdre.

Voici la procédure qui va vous permettre de réaliser ce transfert d'un fichier à l'autre :

```
Proc Struct
ouvre "CLBIS" logique "SECOND"
ouvre "CLIENT" logique "MAITRE"
début
partout
que second.numcl$=maitre.numcl$
que second.societe$=maitre.societe$
que second.nom$=maitre.nom$
que second.prenom$=maitre.Prenom$
que second.adress1$=maitre.adresse$
que second.adress2$=""
que second.cp$=maitre.Cp$
que second.ville$=maitre.Ville$
que second.tel$=maitre.Tel$
que second.solde=maitre.Solde
que second.comment$=""
ajoute "SECOND"
fpartout
ferme
ferme
fproc
```

Cette procédure ouvre les deux fichiers en même temps, ensuite Archive se positionne sur la première fiche grâce à la commande 'Début' et transfert les valeurs du fichier 'CLIENT' au fichier 'CLBIS', après quoi il met à jour l'enregistrement dans le nouveau fichier en mettant les nouvelles rubriques à zéro (que second.comment$=""). Il effectue ce travail sur tout les enregistrements du fichier puis ferme les deux fichiers.
La procédure suivante vous montre comment utiliser le langage d'Archive pour effectuer des restructurations de fichiers mais aussi des mises à jours multiples, vérifications de tout un fichier, etc.

Par exemple vous pourriez avoir besoin d'envoyer un courrier type à tous les clients qui ont un solde positif, alors que vous désirez insérer l'intitulé 'NP' dans la rubrique commentaire pour les autres (Non Productif).

Dans ce cas vous devriez écrire une petite procédure qui va ouvrir votre fichier et passer en revue tous les enregistrements soit pour en imprimer une étiquette, soit pour y ajouter un commentaire (NP).

```
Proc Produc
ouvre "CLIENT" logique "MAITRE"
début
partout
si maitre.solde>0
imprime maitre.societe$
imprime maitre.nom$;" ";maitre.prenom$
imprime maitre.adress1$
imprime maitre.adress2$
imprime maitre.cp$;" ";maitre.ville$
imprime
imprime
imprime
sinon
que maitre.comment$=maitre.comment$+" NP"
```

```
ajoute "MAITRE"
fsi
fpartout
ferme
```

La procédure ci-dessus ne tente pas de démontrer une procédure d'impression d'étiquettes mais l'utilisation de la commande (partout, fpartout), qui permet de passer en revue tous les enregistrements du fichier.

EASEL : QUATRIEME ROUE DU CAROSSE?

Easel est un logiciel graphique très performant, dont la puissance et la simplicité de mise en oeuvre permettent à un simple utilisateur la réalisation de graphes. Il est important d'avoir toujours présent à l'esprit les possibilités d'échanges de données entre Archive et Easel ou entre Abacus et Easel. Ces échanges d'informations font intervenir une double procédure dite 'd'exportation' dans un sens et 'd'importation' dans l'autre sens. Chacun des programmes dispose de ses propres options 'Exporte' et 'Importe' situées dans le menu Documents. Imaginons un tableau Abacus contenant les données suivantes dont vous voulez obtenir une représentation graphique :

	JANVIER	FEVRIER	MARS
Gilles	1000	1500	1700
Alex	500	1100	1500
Phil	990	1800	1100
Fred	750	990	1200

Il vous faut exporter votre document Abacus vers Easel (manuel page 110 et suivantes). Vous obtenez alors un document texte qu'il va être nécessaire d'importer dans Easel (manuel page 243 et suivantes). Toutes ces opérations viennent d'être effectuées manuellement sur votre clavier, mais ne pourrions nous pas automatiser cette opération ?
Le langage TSL (page 255 et suivantes) mis à votre disposition par PC4 permet d'enregistrer sur disque une suite de frappes au clavier et de les restituer à la demande de l'utilisateur. Vous pouvez ainsi imaginez la réalisation de tâches répétitives et souvent peu agréables entièrement automatisées. Le système d'exploitation de votre PC (MS-DOS) permet lui aussi d'automatiser un certain nombre d'opérations (fichiers BAT). Il est naturellement possible de coupler ces deux systèmes pour optimiser et améliorer les performances.

Exemple d'un fichier BAT permettant un export d'Abacus suivi d'un import dans Easel :

```
abacus /tsl=export
easel /tsl=import
```

Le contenu du présent document composé sur Macintosh II et PageMaker donne l'information la plus complète et la plus exacte possible au moment de la publication, mais n'est ni garanti quant à la complétude, ni quant à l'exactitude. **Aware, OMNIS, Quartz, Xchange, Archive, Abacus, Quill, Easel, PC4, Organiseur II, PageMaker, Apple II, IIe, IIgs, Macintosh, ImageWriter, LaserWriter, IBM** sont des marques déposées.

N° de version en cours

ORGANISEUR

CM	3.3
XP	3.6
POS 200	3.6
POS 250	3.6
POS 350	3.6
LZ	4.5
LZ64	4.5
Comms.	4.1
Code Bar	3.2
Carte	2.0
Printer II	1.5
Printer IIc	1.5
Log. 4 Lignes	
Top Fin.	2.2
Filepak	2.0
Tableur	2.0
Prakpak	1.0
Log. 2 Lignes	
Top Fin.	1.3
Filepak	1.4
Barpak	0.9
Tableur	2.0
Prakpak	1.0
Correct.	2.0
Formulat.	1.0
Portfolio	1.0
Travel.	1.5
Maths.	1.0
Fin pak.	1.0
Développ.	2.2
Diary link	1.0
CL PC	2.1
CL MAC	2.1

PC4

PC4	1.1

XCHANGE

Xchange	1.37