PSION

Bulletin Technique

Juillet Août Septembre 1989

La revue du support technique d'Aware

ORGANISEUR II

PC4

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

OPL langage de l'année !

La toute nouvelle année se doit de commencer par de bonnes résolutions ; chez Aware, nous ne ferons pas exception à la règle ; voici sans plus attendre le programme des réjouissances :

Afin de faciliter encore les contacts entre les utilisateurs des produits 'AWARE'et leur support technique, nous vous annonçons la mise en place du service minitel gratuit "36 16 AWARE", qui, nous l'espèrons, vous rendra les plus grands services.

Programmeurs, développeurs, vous allez pouvoir débrider votre imagination avec l'arrivée des nouveaux MC : autonomes, performantes, ces nouvelles machines sont conviviales (zone tactile, boutons radio, boites à cocher...) et, qui plus est, programmables en OPL.

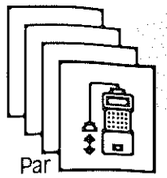
Sollicités par de très nombreux utilisateurs , les nouveaux RamPacks 128k et 256k vous offriront bientôt tout l'espace mémoire dont vous avez besoin.

Enfin les bulletins techniques seront désormais plus réguliers (le support technique en a fait le serment le premier janvier à minuit une...)

Nous vous souhaitons à tous une excellente année et de bons programmes 1990.

Philippe DELAPLACE

OPL ET SES FICHIERS



Par
Gilles JEAN

Le concept de fichier apparaît dans toutes les applications de gestion. Par fichiers, on entend une collection d'informations pouvant être consultées individuellement de façon itérative et systématique. Un fichier constitué d'une série d'enregistrements peut être stocké sur un support tel un Datapak, un Rampak ou la mémoire vive de votre Organiseur.
Voici la liste des commandes et fonctions que nous allons utiliser dans l'exemple qui suit.

```
CREATE
Syntaxe: create"dev:fname",log,rub1,rub2...rub16
Exemple: create"b:articles",a,code$,desc$,pa,pv,qte
```

Cette commande permet de créer un fichier sur l'unité <dev>, sous le nom <fname>. Il porte le nom logique <log> et peut contenir au maximum 16 champs différents (rubrique). Le nom logique d'un fichier doit être A,B,C ou D et il peut y avoir jusqu'à quatre fichiers ouverts en même temps mais sous des noms logiques différents. Lorsqu'un fichier est créé, il est automatiquement considéré comme ouvert.

```
OPEN
Syntaxe: open"dev:fname",log,rub1,rub2...rub16
Exemple: open"b:articles",a,code$,desc$,pa,pv,qte
```

Cette commande permet d'ouvrir un fichier sur l'unité <dev>, sous le nom <fname>. Il porte le nom logique <log> et peut contenir au maximum 16 champs différents (rubrique). Le nom logique d'un fichier doit être A,B,C ou D et il peut y avoir jusqu'à quatre fichiers ouverts en même temps mais sous des noms logiques différents. Un fichier peut être ouvert avec moins de rubriques qu'il n'a été créé.

```
USE
Syntaxe: use <log>
Exemple: use a
```

Cette commande permet de sélectionner le fichier ayant été créé ou ouvert avec le nom logique <log>.

```
CLOSE
Syntaxe: close <log>
Exemple: close a
```

Cette commande permet de fermer le fichier ayant été ouvert ou créé avec le nom logique <log>. Il est nécessaire d'avoir positionné le fichier en vigueur (voir commande use) avant de le fermer.

```
DELETE
Syntaxe: delete"dev:fname"
Exemple: delete"a:articles"
```

Cette commande permet de supprimer un fichier sur l'unité <dev> portant le nom <fname>. Le fichier doit avoir été fermé avant que cette commande ne soit exécutée. Tout les enregistrements de ce fichier seront détruits.

```
COPY
Syntaxe: copy"dev1:fname1","dev2:fname2"
         copy"dev1:fname1","dev2:"
         copy"dev1:","dev2:"
Exemple: copy"a:articles","b:backart"
         copy"a:articles","b:"
         copy"a:","b:"
```


Cette commande permet de copier un/des fichier(s) sur une unité différente. Si l'unité de destination contient déjà un fichier du même nom <dev2:fname2>, les enregistrements de l'unité source <dev1:fname1> sont rajoutés à la fin de ce fichier. Dans le premier exemple, le fichier <a:articles> est copié sur l'unité <b:> sous le nom <baclart>. Dans le deuxième exemple, le fichier <a:articles> est copié sur l'unité <b:> sous son nom d'origine. Dans le troisième exemple, tout les fichiers de l'unité <def1> sont copiés sur l'unité <def2> sous leur nom d'origine.

```
RENAME
Syntaxe: rename<dev:fname1>,<dev:fname2>
Exemple: rename"a:articles","a:artold"
```

Cette commande permet de renommer le fichier <fname1> sur l'unité <dev> en <fname2>. Il est important de noter que l'unité <dev> ne doit pas être modifiée.

Exemple n°1 (1 procédure)

Cet exemple permet d'ouvrir (s'il existe) ou de créer (s'il n'existe pas) un fichier avec une seule rubrique (a.a$). Un message apparaît alors signalant le travail qui vient d'être effectué.

```
test1:
local fnom$(10)
do
cls
print"Nom du fichier?"
input fnom$
until len(fnom$)
cls
if exist(fnom$)
open fnom$,a,a$
print"Ok Ouvert..."
else
create fnom$,a,a$
print"Ok creer..."
endif
beep 99,99
print"Tapez sur space"
get
```

Exemple n°2 (1 procédure)

Cet exemple permet d'ouvrir (s'il existe) ou de créer (s'il n'existe pas) un fichier avec une seule rubrique (a.a$). L'intérêt d'une telle procédure par rapport à celle figurant ci-dessus est que vous utiliserez moins de place en mémoire vive lors de l'exécution et bien sûr moins d'espace de stockage (Datapak/Rampak).

```
test2:
local fnom$(10)
do
cls
print"Nom du fichier?"
input fnom$
until len(fnom$)
cls
trap open fnom$,a,a$
trap create fnom$,a,a$
beep 99,99
print"Tapez sur space"
get
```

Exemple n°3 (2 procédures)

Cet exemple permet d'ouvrir (s'il existe) ou de créer (s'il n'existe pas) un fichier avec une seule rubrique (a.a$). L'intérêt d'un telle procédure par rapport à celles figurant ci-dessus est que vous contrôlerez en totalité la saisie de l'utilisateur. Un petit inconvénient sera, contrairement aux autres procédures, d'être plus gourmand en espace mémoire lors de l'utilisation. Notez toutefois que ces procédures auraient pu être un excellent support pour un article traitant le procédural et le transfert de paramètres du langage OPL de l'Organiseur II.

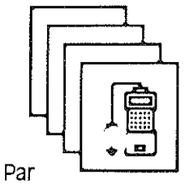
```
test31:
global n$(8),p$(1)
p$="A"
if test32:("Nom du fichier ?")
trap create p$+":"+n$,a,a$
trap open p$+":"+n$,a,a$
endif

test32:(c$)
local p%,k%
cls
print c$
do
kstat 1
at 1,2
print chr$(15);
print left$(p$+":"+n$,10)
at 3+p%,2
cursor on
k%=get
if k%=1
if n$<>""
n$=""
p%=0
else
return 0
endif
elseif k%=2
do
if p$="A"
p$="B"
elseif p$="B"
p$="C"
else
p$="A"
endif
until exist(p$+":main")
elseif k%=3
p%=0
elseif k%=4
p%=len(n$)
elseif k%=5 and p%
p%=p%-1
elseif k%=6 and p%<len(n$)
p%=p%+1
elseif k%=7 and p%<len(n$)
n$=left$(n$,p%)+mid$(n$,p%+2,9)
elseif k%=8 and p%
n$=left$(n$,p%-1)+mid$(n$,p%+1,9)
p%=p%-1
elseif k%>32
if len(n$)=8
beep 99,99
else
n$=left$(n$,p%)+chr$(k%)+mid$(n$,p%+1,9)
p%=p%+1
endif
endif
until k%=13 and len(n$)
return 1
```

Exemple n°4

Cet exemple permet de détruire un fichier de données, de le copier ou de le renommer. Attention, cet utilitaire ne gère absolument pas les erreurs pouvant survenir lors de son utilisation et sortira automatiquement du programme en affichant le message d'erreur standard.

```
test4:
global n$(8),p$(1)
local n2$(8),p2$(1)
local m%
p$="A"
do
m%=menu("Detruire,Copier,Renommer")
if m%=1
if test32:("Detruire ?")
delete p$+":"+n$
endif
elseif m%=2
if test32:("Copier source ?")
p2$=p$
n2$=n$
if test32:("Copier destin ?")
copy p2$+":"+n2$,p$+":"+n$
endif
endif
elseif m%=3
if test32:("Renommer ?")
cls
print"Nouveau nom ?"
input n2$
rename p$+":"+n$,n2$
endif
endif
until m%=0
```

Il ne vous reste plus qu'a implémenter ces routines dans vos programmes afin d'obtenir un meilleur contrôle 'utilisateur' (exemple 3) ou une occupation moindre de votre mémoire (exemple 1 & 2).



Par
Frédéric MARCHESI

TESTER L'ETAT DE LA TOUCHE "SHIFT"

L'Organiseur ne possède que le nombre nécessaire de touches pour ne pas avoir un clavier trop chargé, ce qui vous amène sûrement à manquer de touches de fonction dans vos programmes. Toutes les touches ayant déjà une utilisation bien définie, il ne nous reste plus que la touche "Shift" qui ne correspond à aucune fonction particulière si ce n'est de permettre de changer le mode de clavier (Majuscule, Minuscule, Numérique).

Vous pouvez remarquer que la touche "Shift" n'a pas le même effet que les autres par le simple fait qu'en frappant cette touche vous n'entendez pas

le petit Bip habituel, donc on ne peut pas tester cette touche comme on teste les autres avec les commandes (Get, View, Disp ou Key).

Key : Cette commande permet de transférer la valeur de la dernière touche frappée à une variable numérique sans être stoppé sur la ligne en question, ce qui nous donne la possibilité pendant cette boucle de glisser un petit test sur l'état de la touche "Shift".

Tapez les deux procédures ci-dessous :

```
CTRL:
GLOBAL A%,LIGNE$(16),S%
ESCAPE OFF
S%=PEEKB($63)
DO
A%=KEY
AT 1,1 :PRINT REPT$(" ",4);RIGHT$(DATIM$,8);REPT$(" ",4)
AT 1,2 :PRINT LIGNE$;REPT$(" ",16-LEN(LIGNE$))
IF A%=8
IF LEN(LIGNE$)>0
LIGNE$=LEFT$(LIGNE$,LEN(LIGNE$)-1)
ELSE
BEEP 99,99
ENDIF
ELSEIF A%=31 AND A%<255
IF LEN(LIGNE$)<16
LIGNE$=LIGNE$+CHR$(A%)
ELSE
BEEP 99,99
ENDIF
ELSEIF PEEKB($63)<>S%
LISTE:
S%=PEEKB($63)
ENDIF
UNTIL A%=13

LISTE:
DISP(1,"1 - Majuscule"+CHR$(9)+"2 - Minuscule"+CHR$(9)+"3 - Numérique")
IF PEEKB($75)=17 :KSTAT 1
ELSEIF PEEKB($75)=2 :KSTAT 2
ELSEIF PEEKB($75)=12 :KSTAT 3
ENDIF
RETURN
```

Pour démarrer ce petit programme, vous devez lancer "CTRL". Vous voyez apparaître sur la première ligne l'heure système, la deuxième étant réservée à la saisie. Si vous pressez la touche, "Shift" vous verrez apparaître une liste vous permettant de selectionner le "KSTAT" desiré (Majuscule, Minuscule, Numérique). Faîtes votre choix en pressant les touches "1, 2 ou 3". Pressez la touche "On/Clear" pour sortir de la procédure.

La procédure "CTRL" commence par retenir la valeur stockée à l'adresse "$63" en hexadécimal (S%=PEEKB($63)). A cette adresse est stockée la valeur qui définit au système l'état de la touche "Shift". Ensuite elle effectue une boucle sur le contrôle de saisie avec la commande "KEY", ce qui permet à tout moment de savoir si la valeur de l'adresse "$63" à changé ou non (ELSEIF PEEKB($63)<>S%). La procédure "LISTE" n'est appelée que si ces deux valeurs sont différentes.

La procédure "LISTE" affiche dans un "DISP" les différents modes de clavier possibles. Une adresse est utilisée dans cette procédure, elle contient la valeur de la dernière touche frappée "$75".

		Maj	Min	Num
La touche "U" a la valeur 17	ASCII	85	117	49
La touche "V" a la valeur 2	ASCII	86	118	50
La touche "W" a la valeur 12	ASCII	87	119	51

Nous effectuons les tests de la pression des touches sur leur valeur physique et non sur leur valeur ASCII. Ceci nous évite d'avoir à tester trois valeurs correspondant à une même touche.

Par exemple :

```
test1:
local a%
a%=view(1,"Pressez la touche Q")
if a%=81 or a%=54 or a%=113
...
endif

test2:
view(1,"Pressez la touche Q")
if peekb($75)=13
...
endif
```

La procédure "test2" permet de ne pas déclarer de variables. De plus nous n'avons plus qu'une condition dans notre test (if peekb($75)=13), 13 étant la valeur que retourne la touche "Q". La procédure la plus courte sera la plus rapide donc la plus performante. Mais attention cette solution n'a d'intérêt que si quelques touches seulement sont utilisées.

Dans les Bulletins Techniques précédents, nous avons décrit un bon nombre de techniques permettant d'effectuer des contrôles de saisie où l'on gère la montée et la descente des lignes dans une liste de rubriques. Un moyen simple permet d'écourter un peu ces procédures en utilisant l'adresse nous donnant l'état de la touche "SHIFT", ce qui nous permet d'utiliser la commande "EDIT".

Saisissez la procédure ci-dessous :

```
SAISIE:
LOCAL RUB$(11,10),L$(11,25),X,S%
REM *** S% = L'etat de la touche shift :X = Numero de ligne ***
S%=PEEKB($63) :X=1
REM *** RUB$(X) = intitule des rubriques ***
RUB$(1)="Nom :"
RUB$(2)="Prenom :"
RUB$(3)="Tel :"
RUB$(4)="Telex :"
RUB$(5)="Fax :"
RUB$(6)="Addr :"
RUB$(7)="Code PTT :"
RUB$(8)="Ville :"
RUB$(9)="Pays :"
RUB$(10)="Code :"
CLS
REM *** Depart de la boucle avec test X<11 ***
WHILE X<11
AT 1,2
REM *** Test de la deuxieme ligne d'affichage ***
IF X<10 AND LEN(RUB$(X+1)+L$(X+1))<16
PRINT RUB$(X+1);L$(X+1);
PRINT REPT$(" ",16-(LEN(RUB$(X+1))+LEN(L$(X+1))))
ELSEIF X+1<10 AND LEN(RUB$(X+1)+L$(X+1))>16
PRINT RIGHT$(RUB$(X+1)+L$(X+1),16)
```

```
ELSEIF X=10
PRINT REPT$(" ",16)
REM *** Fin des test de la deuxieme ligne d'affichage ***
ENDIF
REM *** Impression de la premiere ligne d'affichage ***
AT 1,1 :PRINT RUB$(X);
REM *** Edition de la variable en question ***
EDIT L$(X)
REM *** Test de la sortie de l'EDIT soit "EXE" ou "Shift EXE" ***
REM *** Sortie EXE :S% toujours egale a la valeur peekb($63) ***
IF PEEKB($63)=S% AND X<11 :X=X+1
REM *** Sortie Shift EXE :S% different de la valeur a l'adresse $63 ***
ELSEIF PEEKB($63)<>S% AND X>1 :X=X-1
REM *** Fin des tests ***
ENDIF
REM *** Fin de la boucle ***
ENDWH
```

Cette procédure donne la possibilité de saisir des informations dans une liste de rubriques.
Attention : Cette procédure permet de saisir mais ne met pas à jour dans un fichier. Il faudra ensuite créer une procédure de mise à jour.

Vous n'êtes pas obligé de saisir les lignes de commandes commençant par "REM". Ces lignes ne sont là que pour vous permettre de bien suivre le déroulement de la procédure.

Avec cette procédure, vous pouvez saisir sur 10 lignes, différents renseignements comme le nom, le prénom, le tel, etc.

EXE passe à la ligne suivante
Shift EXE passe à la ligne précédente
Si vous arrivez à la dernière ligne et que vous frappez seulement la touche EXE, vous sortez du programme.

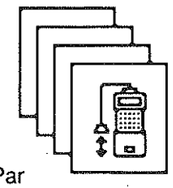
Si vous désirez connaître les valeurs des touches qui se lisent à l'adresse $75, il suffit de faire une petite procédure qui va mettre en attente de l'appui d'une touche, pour pouvoir ensuite afficher le "PEEKB($75)".

```
TCH:
LOCAL A%
DO
A%=GET
AT 1,1 :PRINT"ASCII       :";CHR$(A%);" "
AT 1,2 :PRINT"PEEKB($75)  :";PEEKB($75);" "
UNTIL A%=1
```

Chaque fois que vous presserez une touche, l'Organiseur vous affichera :
- sur la première ligne, le caractère en question,
- sur la deuxième ligne, la valeur stockée à l'adresse $75.
La touche "On/Clear" vous permet de quitter cette procédure.

UNE NOUVELLE COMMANDE D'AFFICHAGE EN OPL

Le langage Opl possède une commande qui permet de faire défiler un message de 255 caractères maximum sur l'écran "VIEW" .L'inconvénient c'est que l'on est forcé de faire défiler la ligne en entier, il pourrait être



Par
Frédéric MARCHESI

pratique de n'en faire défiler qu'une petite partie.

La procédure ci-dessous vous permet de définir la zone dans laquelle vous désirez faire défiler votre message.

```
view:(t$,l%,d%,g%,t%)
local x%,s%,q%,b%
pokeb $0077,5 :pokew $20CB,0
onerr bug::
bug::
s%=1 :x%=1 :b%=b%+1
if len(t$)>l%
if l%<=20-d% and d%<=20-l% and g%>=1 and g%<=4
do
pause -(peekw($69)+1*(abs(peekw($69)<=0)))
q%=key
at d%,g% :print mid$((t$+"    "+t$),x%,l%)
if q%=5
if s%=1 :s%=3
elseif s%=3 :s%=2 :endif
elseif q%=6
if s%=2 :s%=3
elseif s%=3 :s%=1 :endif :endif
if s%=1 :x%=x%+1
elseif s%=2 :x%=x%-1 :endif
if x%=len(t$)+4 and s%=1 :x%=1
elseif x%=1 :x%=len(t$)+4 :endif
if t%<>0 and t%<peekw($20CB)
b%=11 :endif
until q%<>0 and q%<>5 and q%<>6 or b%>10
endif
else :at d%,g% :print t$
q%=get :endif
pokeb $0077,$0E
return q%
```

Cette procédure ne s'utilise pas comme un programme elle doit être utilisée comme une commande.

Par exemple :

```
local a$(28)
a$="Test de defilement a l ecran"
view:(A$,10,6,2,0)
```

cet exemple Fait défiler le message dans la zone définie

VIEW:(Texte,Long,Debut,Ligne,Temps)

Texte:	Message à faire défiler entre guillemets ou sous forme de variable
Long:	Longueur de la zone à faire défiler à l'écran
Debut:	Position ou commence le message défilant
Ligne:	Numéro de ligne ou doit défiler le message
Temp:	Temps de défilement du message en 20ème de secondes
	Cette valeur mise à 0 permet un défilement sans notion de temps

Cette commande vous permet comme la commande 'VIEW' de l'Opl, de connaître la valeur ASCII de la touche qui a fait passer l'Organiseur à la commande suivante.

Par exemple :

```
Test1:
local a$(50),nbl%,tch%
nbl%=1
do
a$="Saisissez le caractere No "+gen$(nbl%,2)
tch%=view:(a$,10,6,2,200)
if tch%>=31 and tch%<=255
nbl%=nbl%+1
at 10,4
print chr$(tch%)
endif
until tch%=13
```

Cet exemple permet d'avoir un message défilant sur une partie précise de votre écran, tout en testant la touche qui fait passer à la commande suivante, pour cela vous devrez déclarer une variable du type numérique.

Dans l'exemple précédent on utilise la variable 'tch%' pour connaitre la valeur ASCII de la touche qui vous à fait passer à la commande suivante, une liste des valeurs ASCII de tout les caractères figure dans le manuel de l'Organiseur.

Les valeurs définies pour 'VIEW:' dans l'exemple précédent indique à l'Organiseur que vous désirez avoir un message défilant (a$), qui défilera dans une zone de 10 caractères en commençant sur le sixième de la deuxième ligne pendant 5 secondes (200).

Cette commande réagit exactement comme la commande 'VIEW' de l'Opl, si vous pressez une touche autre que les flèches gauche ou droite la variable numérique utilisée avec 'VIEW:' vous retourna la valeur de la touche frappée sinon si aucune touche n'est frappée et que le temps défini est passé la variable prendra la valeur 0.

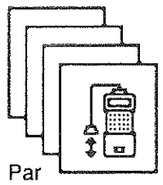
Le programme présenté dans cet article à été réalisé sur le nouveau modèle de PSION possédant 4 lignes d'affichage (LZ ou LZ64), donc si vous possèdez un modèle n'affichant que sur deux lignes (XP ou CM), il vous faut modifier une partie du programme pour l'adapter sur votre modèle, de façon à ne pas pouvoir demander l'affichage de votre message sur plus de deux lignes de 16 caractères.
La modification à effectuer est simple il suffit de remplacer la ligne :

```
if l%<=16-d% and d%<=16-l% and g%>=1 and g%<=2
```

par la ligne :

```
if l%<=20-d% and d%<=20-l% and g%>=1 and g%<=4
```

Une fois que vous aurez effectué cette modification vous pourrez utiliser cette nouvelle commande sans aucun risque avec votre CM ou XP.



Par
Joël DROUIN

UTILITAIRES FICHIER

Voici un ensemble de procédures qui vous permettra de gérer l'impression d'un fichier. Il vous faudra lancer la procédure principale IMPCAL, ensuite vous devez préciser l'unité sur laquelle se trouve votre fichier et le nom de celui-ci.

Le menu principal apparaît.

```
BLOC CHOIX TRI
EFFACER RECH IMP
QUIT
```

Passons en revue toutes ces options.

BLOC: Celle-ci vous permet d'imprimer votre fichier par bloc. On vous propose d'entrée le numéro de la première fiche à imprimer et le numéro de la dernière. Les numéros des fiches apparaissent durant l'impression. Il faut savoir que si vous désirez imprimer tout un fichier, vous sélectionnez la première fiche et vous mettez pour la dernière fiche un numéro supérieur au nombre de fiches contenues dans votre fichier.

CHOIX: Vous avez ici la possibilité de choisir les fiches que vous désirez imprimer. Vous choisissez dans un premier temps avec les curseurs monter/descendre la fiche qui vous intéresse, vous travaillez à ce moment-là uniquement sur la première rubrique de cette fiche. Si vous désirez plus de détails sur cet enregistrement, vous appuyez sur EXE pour visualiser les différentes rubriques (monter/descendre avec les curseurs). Pour sélectionner définitivement votre fiche, appuyez de nouveau sur EXE. Vous avez à n'importe quel moment la possibilité de revenir au menu principal avec la touche ON. Pour imprimer ce fichier temporaire, voir la fonction IMP.

TRI: Effectue un classement par ordre alphabétique sur la rubrique sélectionnée (1 à 16).

EFFACER: Vous permet d'effacer le fichier en cours, ou le fichier temporaire dont nous reparlerons plus tard.

RECH: cette fonction effectue une impression du fichier en cours, après sélection d'un critère de recherche sur une rubrique particulière. On vous demande dans un premier temps le numéro de la rubrique
(1 à 16). Ensuite vous définissez la chaîne de caractères qui déterminera votre critère de recherche exacte (attention dans un critère de recherche tous les caractères sont pris en considération "espace,virgule, point, etc.").

IMP: Cette option vous permet d'imprimer le fichier temporaire créé avec l'option "CHOIX".

Je vous propose maintenant de vous expliquer le déroulement de ces procédures.
Cette première procédure est la principale. On y ouvre le fichier et l'unité sur lesquels on veut travailler (unit$,nom$) et on vérifie si l'on se trouve sur un fichier FILEPAK (ligne chr$(64)). Ensuite nous avons accès au menu principal par l'intermédiaire duquel nous appelons les sous-procédures.

```
IMPCAL:
GLOBAL W%,FL%,Z%,FLAG,UNIT$(1),NOM$(10)
```

```
GLOBAL A%,B%,C%,D%,E%,F%,G%,U$(1),V%
GLOBAL A$(254),B$(254)
GLOBAL C$(254),D$(254),E$(254),F$(254),G$(254)
GLOBAL H$(254),I$(254),J$(254),K$(254),L$(254),M$(254)
GLOBAL N$(254),O$(254),P$(254)
DO
DO
V%=65
DO
CLS
PRINT CHR$(V%)+":"+NOM$
CURSOR ON
KSTAT 1
U$=GET$
IF U$=CHR$(2)
V%=V%+1
ELSEIF U$=CHR$(1)
IF LEN(NOM$) :NOM$="" :ELSE STOP :ENDIF
ELSEIF U$=CHR$(13)
BREAK
ELSEIF ASC(U$)<33
BEEP 99,99
ELSE
NOM$=NOM$+U$
ENDIF
UNTIL V%=68
UNTIL U$=CHR$(13)
CLS
DO
UNIT$=CHR$(V%)
PRINT UNIT$
PRINT NOM$
GET
UNIT$=CHR$(V%)
TRAP CLOSE
TRAP OPEN UNIT$+":"+NOM$,A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
FIRST
IF LEFT$(A.A$,1)=CHR$(64) :FL%=1
ELSE FL%=0 :ENDIF
E%=MENU("BLOC,CHOIX,TRI,EFFACER,RECH,IMP,QUIT")
IF E%=1 :SELEC:
ELSEIF E%=2 :CHOIX3:
ELSEIF E%=3 :TRI:
ELSEIF E%=4  :DELE:
ELSEIF E%=5 :CRIT:
ELSEIF E%=6 :DIM:
ENDIF
UNTIL E%=7
W%=MENU("NOUV,QUIT")
UNTIL W%=2
```

Voici une des sous-procédures. Nous en avons réalisé neuf. "SELEC" est appelée lorsque l'on selectionne l'option Impression par bloc. La variable a% est utilisée pour pointer sur le premier enregistrement à imprimer et c% nous permet d'incrémenter le compteur des enregistrements à imprimer dans la procédure "IMP".

```
SELEC:
PRINT "PREMIERE FICHE" :INPUT A%
CLS :PRINT "DERNIERE FICHE" :INPUT C%
CLS :POSITION A% :B%=A% :IMP:
```

"IMP" est la procédure d'impression utilisée par les différentes procédures, ("selec","crit","dim").On imprime la rubrique si celle ci est non vide (a.a$<>33 etc...). Z% est égale à 1 si l'impression a été lancée de la procédure "CRIT", flag est égal à 1 si on a ouvert un fichier FILEPAK, ensuite on imprime le numéro de l'enregistrement en cours et on sort de la boucle si b%=c% ou si nous arrivons à la fin du fichier.

```
IMP:
DO :B%=B%+1+FL%
IF A.A$<>"" :LPRINT A.A$ :ENDIF
IF A.B$<>"" :LPRINT A.B$ :ENDIF
IF A.C$<>"" :LPRINT A.C$ :ENDIF
IF A.D$<>"" :LPRINT A.D$ :ENDIF
IF A.E$<>"" :LPRINT A.E$ :ENDIF
IF A.F$<>"" :LPRINT A.F$ :ENDIF
IF A.G$<>"" :LPRINT A.G$ :ENDIF
IF A.H$<>"" :LPRINT A.H$ :ENDIF
IF A.I$<>"" :LPRINT A.I$ :ENDIF
IF A.J$<>"" :LPRINT A.J$ :ENDIF
IF A.K$<>"" :LPRINT A.K$ :ENDIF
IF A.L$<>"" :LPRINT A.L$ :ENDIF
IF A.M$<>"" :LPRINT A.M$ :ENDIF
IF A.N$<>"" :LPRINT A.N$ :ENDIF
IF A.O$<>"" :LPRINT A.O$ :ENDIF
IF A.P$<>"" :LPRINT A.P$ :ENDIF
LPRINT
IF Z%=1 :RETURN :ENDIF
IF FLAG=1 :PRINT B% :ELSE :PRINT B%+FL%-1 :ENDIF
PAUSE 20 :NEXT :CLS :UNTIL B%=C% OR EOF :FLAG=0 :RETURN
B%=0
```

Dans les premières lignes de "CHOIX3", nous gérons les touches des curseurs haut et bas dans une boucle do/until (aa%=3, aa%=4), La première boucle do/until nous permet de scanner la première rubrique de chaque fiche et la seconde met à jour un fichier temporaire par l'intermédiaire de variable chaîne gérée dans les deux procédures "CHANGE1" et "CHANGE2".

```
CHOIX3:
LOCAL A%,AA%,H%,X%
PRINT UNIT$
PRINT NOM$
GET
FIRST :PRINT A.A$
DO :DO :USE A :AA%=GET :CLS
IF AA%=3 :BACK :ELSEIF AA%=4 :NEXT :ENDIF
PRINT A.A$ :X%=POS
IF EOF :PRINT"FIN DE FICHIER" :BEEP 100,200 :ENDIF
UNTIL AA%=13 OR AA%=1
IF AA%=1 :RETURN
ENDIF
AA%=DISP(-1,A$) :IF AA%=1 :RETURN :ELSEIF AA%=13 :ENDIF
A%=VIEW(2,"INSERER (EXE)") :CLS
IF A%=13 :CHANGE1: :ELSE :GOTO RETOUR:: :ENDIF :CLOSE
IF EXIST("A:TAMP2")
OPEN "A:TAMP2",A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
ELSE CREATE "A:TAMP2",A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
ENDIF :CHANGE2: :APPEND :CLOSE
OPEN UNIT$+":"+NOM$,A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
RETOUR::
POSITION X%
PRINT A.A$
UNTIL AA%=1000
RETURN if a%=13 :change1: :else :goto retour:: :endif :close
if exist("a:tamp2")
open"a:tamp2",a,a$,b$,c$,d$,e$,f$,g$,h$,i$,j$,k$,l$,m$,n$,o$,p$
else create"a:tamp2",a,a$,b$,c$,d$,e$,f$,g$,h$,i$,j$,k$,l$,m$,n$,o$,p$
endif :change2: :append :close
open unit$+":"+nom$,a,a$,b$,c$,d$,e$,f$,g$,h$,i$,j$,k$,l$,m$,n$,o$,p$
retour::
position x%
print a.a$
```

```
toto::
until aa%=1000
return
```

Ces deux procédures permettent d'effectuer un transfert d'enregistrements de deux fichiers ayant le même nom logique et ainsi nous pouvons utiliser la procédure d'impression "IMP" pour imprimer le fichier tampon.

```
CHANGE1:
A$=A.A$
B$=A.B$
C$=A.C$
D$=A.D$
E$=A.E$
F$=A.F$
G$=A.G$
H$=A.H$
I$=A.I$
J$=A.J$
K$=A.K$
L$=A.L$
M$=A.M$
N$=A.N$
O$=A.O$
P$=A.P$

CHANGE2:
A.A$=A$
A.B$=B$
A.C$=C$
A.D$=D$
A.E$=E$
A.F$=F$
A.G$=G$
A.H$=H$
A.I$=I$
A.J$=J$
A.K$=K$
A.L$=L$
A.M$=M$
A.N$=N$
A.O$=O$
A.P$=P$
```

Dans cette procédure, nous comparons les valeurs Ascii des caractères de la rubrique sélectionnée pour le tri, en commençant par le dernier enregistrement. Si upper$(a$)<pb$ alors pb$=upper$ donc la valeur Ascii de la variable upper$ devenant inférieure à pb$, celle-ci devient la variable de comparaison ; ce test est réalisé jusqu'au premier enregistrement. Nous avons à ce moment-là le futur premier enregistrement de notre fichier trié et nous rangeons celui-ci à la fin avec un update. Ce cycle continue tant que der%>f %

```
TRI:
LOCAL DER%,F%,PB$(255),A$(254),PB%,LPOS%,F2%,NUM%
FIRST :IF LEFT$(A.A$,1)=CHR$(64) :F%=2 :F2%=1
ELSE :F%=1 AND F2%=0 :ENDIF
PRINT "NUM RUB (1 A 16)"
AT 1,2 :INPUT NUM%
CLS :AT 1,1 :PRINT "REORGANISE..."
LAST :DER%=POS
IF COUNT>0
WHILE DER%>F2%
POSITION DER% :PB%=POS
IF NUM%=1 :A$=A.A$
ELSEIF NUM%=2 :B$=A.B$
ELSEIF NUM%=3 :C$=A.C$
ELSEIF NUM%=4 :D$=A.D$
ELSEIF NUM%=5 :E$=A.E$
ELSEIF NUM%=6 :F$=A.F$
ELSEIF NUM%=7 :G$=A.G$
ELSEIF NUM%=8 :H$=A.H$
ELSEIF NUM%=9 :I$=A.I$
ELSEIF NUM%=10 :J$=A.J$
ELSEIF NUM%=11 :K$=A.K$
ELSEIF NUM%=12 :L$=A.L$
ELSEIF NUM%=13 :M$=A.M$
ELSEIF NUM%=14 :N$=A.N$
ELSEIF NUM%=15 :O$=A.O$
ELSEIF NUM%=16 :P$=A.P$
ENDIF
PB$=UPPER$(A$) :DO
IF NUM%=1 :A$=A.A$
ELSEIF NUM%=2 :B$=A.B$
ELSEIF NUM%=3 :C$=A.C$
ELSEIF NUM%=4 :D$=A.D$
ELSEIF NUM%=5 :E$=A.E$
ELSEIF NUM%=6 :F$=A.F$
ELSEIF NUM%=7 :G$=A.G$
ELSEIF NUM%=8 :H$=A.H$
ELSEIF NUM%=9 :I$=A.I$
ELSEIF NUM%=10 :J$=A.J$
ELSEIF NUM%=11 :K$=A.K$
ELSEIF NUM%=12 :L$=A.L$
ELSEIF NUM%=13 :M$=A.M$
ELSEIF NUM%=14 :N$=A.N$
ELSEIF NUM%=15 :O$=A.O$
ELSEIF NUM%=16 :P$=A.P$
ENDIF
IF UPPER$(A$)<PB$
PB$=UPPER$(A$) :PB%=POS :ENDIF
LPOS%=POS :BACK
UNTIL POS=F% AND LPOS%=F%
POSITION PB%
AT 1,2 :PRINT CHR$(15);LEFT$(PB$,16); :UPDATE :DER%=DER%-1
ENDWH
ENDIF
```

Cette procédure vous permettra d'effacer le fichier en cours, ou le fichier temporaire.

```
DELE:
LOCAL F%,DEL%
DEL%=MENU("FICHIER,TAMPON")
F%=VIEW(1,"EFFACER O/N")
IF F%=78 OR F%=110 :RETURN :ENDIF
IF DEL%=2
IF EXIST("A:TAMP2")
CLOSE
DELETE "A:TAMP2"
ELSE :PRINT"FIC INEXISTANT"
GET
ENDIF
ELSEIF DEL%=1
CLOSE
DELETE UNIT$+":"+NOM$
IF UPPER$(NOM$)="MAIN"
CREATE UNIT$+"MAIN",A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
ENDIF
ENDIF
RETURN
```

Crit est la procédure qui permet de rechercher une fiche grâce à un critère de recherche.

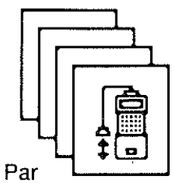
```
CRIT:
LOCAL NUM%,A%,B$(4),CRIT$(254),RUB$(16,254)
PRINT"NUMERO RUBRIQUE"
INPUT NUM%
CLS
PRINT"CRIT DE RECH"
INPUT CRIT$
CLS
FIRST
DO
CLS
B%=FIND(CRIT$)
RUB$(1)=A.A$
RUB$(2)=A.B$
RUB$(3)=A.C$
RUB$(4)=A.D$
RUB$(5)=A.E$
RUB$(6)=A.F$
RUB$(7)=A.G$
RUB$(8)=A.H$
RUB$(9)=A.I$
RUB$(10)=A.J$
RUB$(11)=A.K$
RUB$(12)=A.L$
RUB$(13)=A.M$
RUB$(14)=A.N$
RUB$(15)=A.O$
RUB$(16)=A.P$
DO
A%=A%+1
IF RUB$(A%)=CRIT$ AND A%=NUM%
Z%=1
B%=0
IMP:
ENDIF
UNTIL A%=16
A%=0
NEXT
UNTIL EOF
RETURN
```

Cette dernière procédure permet de vérifier l'existence du fichier temporaire avant son impression.

```
DIM:
CLOSE
IF EXIST("A:TAMP2")
OPEN"A:TAMP2",A,A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$,O$,P$
FLAG=1
IMP:
CLOSE
ELSE
VIEW(1,"PAS DE FICHIER")
BEEP 100,200
ENDIF
RETURN
```

EFFACER LE BUFFER DE COMMS LINK

Dans votre manuel de Comms Link, il est indiqué que la fonction LINPUT$: a la particularité de vider le buffer de réception sous Comms Link lorsqu'elle est munie du paramètre 0 (c'est-à-dire LINPUT$:(0).

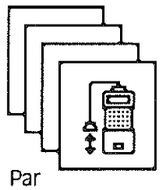


Par
Pierre LAGRANGE

Il semblerait que cela ne soit pas systématiquement le cas. Nous vous proposons alors une petite procédure dont le rôle est justement de vider le buffer de réception :

```
BUFF:
REM effacement buffer
ONERR jnk::
WHILE 1
  JNK::
  a$=LINPUT$:(254,1)
  IF a$="" :BREAK :ENDIF
ENDWH
ONERR OFF
```

Cette procédure vide en effet tous les caractères présents dans le buffer de réception de l'Organiseur. Il fait cela en assignant les caractères du buffer dans la variable a$, puis en remplaçant cette valeur par une autre série de caractères. Lorsqu'il n'y a plus de caractères dans le buffer, une chaîne vide est assignée à a$, la boucle est alors interrompue et la section suivante de la procédure est exécutée.



Par
Gilles JEAN

EMULATION DE L'OPTION VILLE

Heureux possesseurs d'un Organiseur II modèle LZ et LZ64, cet article vous intéresse. Vous n'avez certainement pas résisté à l'envie d'utiliser l'option VILLE qui vous permet de consulter l'indicatif téléphonique et l'heure des grandes villes du monde à partir de votre ville de départ. Vous avez donc pu constater l'aisance et la simplicité avec laquelle vous retrouver une ville en saisissant les trois ou quatre premiers caractères du mot.

Le programme listé ci-dessous vous permet obtenir la même interface utilisateur pour la recherche de vos enregistrements dans un fichier que vous aurez vous-même constitué.

```
ville:
local k%,p%,f%,b$(15)
open"a:art",a,a$,b$
debut::
do
if f%
at 1,1
print chr$(14);"PAS D'AUTRES DONNEES";chr$(15)
f%=0
else at 1,1
print chr$(14);left$(a.a$,20)
at 1,2
print chr$(15);left$(a.b$,20)
endif
at 1,3
print rept$("-",20);
at 1,4
print chr$(23);"Rech:";b$
at 6+p%,4
cursor on
k%=get
```

```
if k%=1
if b$<>""
b$=""
p%=0
else
close
return
endif
elseif k%=3
p%=0
elseif k%=4
p%=len(b$)
elseif k%=5 and p%
p%=p%-1
elseif k%=6 and p%<len(b$)
p%=p%+1
elseif k%=7 and p%<len(b$)
b$=left$(b$,p%)+mid$(b$,p%+2,15)
elseif k%=8 and p%
b$=left$(b$,p%-1)+mid$(b$,p%+1,15) :p%=p%-1
elseif k%>31
if len(b$)<=15
b$=left$(b$,p%)+chr$(k%)+mid$(b$,p%+1,15)
p%=p%+1
endif
endif
if k%<31
first
endif
while left$(a.a$,p%)<>b$
next
if eof
f%=1
goto debut::
endif
endwh
until k%=13 and len(b$)
close
```

Il est important de noter que le fichier sur lequel nous travaillons et effectuons nos recherches, doit avoir été préalablement trié sur la rubrique de référence. Vous pouvez pour ce faire utiliser la routine figurant dans ce numéro (page 14) ou la fonction de tri du menu XFICH de votre Organiseur II modèle LZ ou LZ64.

TABLEAU POUR FICHIERS ARCHIVE

Par
le support technique

Ne vous est-il jamais arrivé de vouloir visualiser vos fichiers Archive (*.dbf) sous forme de tableau ?
Voici un programme qui va vous permettre de réaliser votre voeu.
Tout d'abord, nous allons créer un petit fichier pour notre exemple.

```
proc CREATION
     crée "PRODUIT.DBF"
          CODEP$
          DESIG$
          MATUT$
          PVENT
          fcrée
     fproc
```


Maintenant, insérons quelques fiches (environ 30 pour obtenir au moins deux tableaux de visualisation).

```
proc INSERT
     ouvre "PRODUIT.DBF"
     insère
     saisis "Autre fiche O/N : "; AF$
     si AF$="O": ferme :INSERT: fsi
     ferme
     fproc
```

Ouf ! C'est fait. Passons sans plus attendre au vif du sujet.
Créons un en-tête à notre tableau afin que celui-ci soit lisible lors de sa visualisation. Cet en-tête sera affiché en haut de chaque tableau.

```
proc ENTETE
     écris au 1,1; encre 2;"CODE PRODUIT"
     écris au 1,21; encre 2;"DESIGNATION"
     écris au 1,46; encre 2;"MAT. UTILISABLE"
     écris au 1,66; encre 2;"PRIX VENTE HT"
     fproc
```

La procédure START va paramètrer l'affichage en MODE 0, pour que nous puissions utiliser la taille maximum de l'écran, c'est-à-dire sans le guide. Ensuite elle va ouvrir le fichier, effacer les index générés par les tris, trier notre fichier par ordre croissant sur le code produit, positionner le pointeur sur le premier enregistrement et afficher l'en-tête de notre tableau.
Le fait d'appeler cette procédure START, va nous permettre d'exécuter notre programme automatiquement en tapant : EXEC "TABLEAU", ceci nous évite des commandes du type : CHARGE"TABLEAU" puis START.

```
proc START
     mode 0
     éponge
     ouvre "PRODUIT.DBF"
     efface index
     trie CODEP$;C
     que C=1: que L=3
     début
     ENTETE
     TAB01
```

TAB01 positionne les enregistrements dans notre tableau (L=N° de ligne et C=N° de colonne). La variable numérique PVENT est affichée sous la forme : écris au L,C;déci(PVENT,2,10), afin que celle-ci soit justifiée sur la droite avec deux décimales. La fonction fdf, teste la fin de fichier. Elle est égale à 1 si la fonction est logiquement vraie ou 0 si la fonction est logiquement fausse. Dans le cas où elle est égale à 1 nous allons afficher en bas de l'écran : "FIN DE FICHIER" ; dans le cas contraire, nous lançons la procédure TAB02.

```
proc TAB01
     que C=1
     écris au L,C;CODEP$
     que C=C+20
```

```
      écris au L,C;DESIG$
            que C=C+25
      écris au L,C;MATUT$
      que C=C+20
      écris au L,C;déci(PVENT,2,10)
      suiv
      si fdf()=1
            écris au 23,0; encre 1;"FIN DE FICHIER...";
            écris au 23,20; encre 2;"TAPER UNE TOUCHE POUR CONTINUER"
            que CARAC$=clavier()
            ferme : éponge
            stoppe
            fsi
      TAB02
      fproc
```

La dernière procédure TAB02 teste le nombre de lignes affichées à l'écran. Lorsque que celui-ci est égal à 20, elle vous demande de taper une touche pour afficher le reste du fichier.

```
proc TAB02
      que L=L+1
      si L=20
            que L=23: que C=20
            écris au L,C; encre 2;"TAPER UNE TOUCHE POUR CONTINUER"
            que CARAC$=clavier()
            éponge
            ENTETE
            que L=3
            TAB01
            fsi
      TAB01
      fproc
```

Vous pourrez aussi utiliser ce programme pour obtenir des états. Pour cela vous avez deux solutions :

- soit en utilisant la commande VIA IMPRIMANTE ,
- soit en remplaçant les commandes Ecris par Imprime .

Sachez tout de même que sur une feuille au format A4 vous bénéficiez d'environ 70 lignes. Modifiez donc la ligne si L=20 , qui se trouve dans la procédure TAB02, par si L=60. Voilà, c'est terminé, il ne reste plus qu'à sauvegarder tout cela sous le nom de : TABLEAU.PRG.

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
Bar	3.2
Carte	2.0
Printer II	1.5
Printer IIc	1.5
Top Fin.	1.1
Filepak	1.4
Barpak	0.9
Tableur	2.0
Correct.	2.0
Formulat.	1.0
Portfolio	1.0
Travel.	1.0
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