OPSION

Bulletin Technique

AVRIL MAI JUIN 1990

La revue du support technique d'Aware

MOBILE COMPUTER

- Le MC et l'Editeur de texte
- Fonction Lier du MC

ORGANISEUR II

- Compression de fichiers
- Encore un chronomètre
- L'Organiseur et les dates
- Gestion d'une table d'index
- Transfert de fichiers

PC4

- Initiation à la programmation
  Archive

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

Chaud l'été... Chaud !

Le chaud soleil du mois d'août, le chant des grillons, la peau hâlée ou le déferlement des vagues sur le sable... j'espère que vous profitez pleinement de ces instants pour reprendre des forces et pouvoir déborder de dynamisme à la rentrée.

A Paris en revanche, le bithume fond, on parle toutes les langues, et si le soleil est au rendez-vous, il est aussi très cruel quand il nous tend les bras à travers la fenêtre... Rassurez-vous, nous n'avons pas succombé à son charme et voici le produit de notre travail : le deuxième Bulletin Technique promis, celui qui vous prouve que nous mettons les bouchées doubles !

Dans ce numéro, vous découvrirez un peu plus le Mobile Computer, encore et toujours de nombreux articles sur l'Organiseur II, et notamment un programme sur la Compression de fichiers qui devrait en intéresser plus d'un. Enfin, nous n'oublions pas les utilisateurs de PC4 !

Parallèlement, nous préparons activement la rentrée et je peux d'ores et déjà vous dire qu'elle sera riche en événements. Traditionnellement, nous serons présents à Apple Expo qui aura lieu au Cnit du 19 au 22 septembre. Nous y présenterons le Mobile Computer et l'Organiseur II acoquinés avec le Macintosh, ainsi que les dernières nouveautés dans les deux gammes. D'autres surprises suivront, mais vous n'en saurez pas plus pour le moment...

Bonne rentrée et à bientôt !

Martine Carrigues

Le MC et l'Editeur de textes

Par
Alexandre BOUILLOT

Vous avez probablement déjà remarqué certaines petites "bizarreries" lors de l'utilisation de l'éditeur de textes du MC. Pourtant, chacune de ces opérations qui vous paraissent étranges dispose d'une explication tout à fait logique et rationnelle que nous allons examiner ci-après.

Tout d'abord, voici quelques rappels sur ce qu'est un éditeur de textes. Contrairement à un traitement de textes, l'éditeur ne fait généralement pas la différence entre ligne et paragraphe. Bien souvent, il ne sait pas ce qu'est un paragraphe. Il ne gère que des lignes. Celles-ci sont séparées par un retour chariot et un saut de ligne que l'on appelle plus communément CR LF. Il s'agit de deux codes de contrôles qui ont pour valeur respective 13 et 10. Ces deux codes sont générés dans l'éditeur par la seule pression de la touche de validation ou touche Enter.

L'éditeur de texte du MC étant appelé à gérer des lettres, des mémos etc..., Psion l'a agrémenté d'un certain nombre d'options qui permettent de réaliser ce type de documents. Ainsi, le changement de ligne étant défini par un CR LF, le changement de paragraphe est, lui, défini par un double CR LF, c'est-à-dire une ligne blanche entre deux paragraphes. L'éditeur de textes du MC est également capable de faire des sauts de lignes lorsque l'on arrive ou bout de la ligne, à la marge droite. Mais ces retours ne sont placés que dans les cas suivants : lorsque le MC sait que ce paragraphe est géré automatiquement, lorsqu'il n'y a pas d'exception dans le formattage du paragraphe, ou lorsque l'utilisateur n'insère pas volontairement un retour chariot dans son paragraphe. Dans le cas contraire, l'éditeur va supposer que l'utilisateur préfère faire lui-même sa mise en forme et donc, qu'il n'a pas à intervenir. Aussi, plutôt que d'insérer un retour chariot, l'éditeur va faire "fuir" le texte vers la droite jusqu'à la limite de 235 caractères par ligne.

Après avoir examiné la théorie, voici quelques "truc" sur l'utilisation de l'éditeur de textes du MC.

Comment remettre en forme un paragraphe ?

Après avoir changé de fonte ou de marge à droite, les lignes ne suivent pas la modification. Pour refaire coïncider les lignes et la marge droite, il suffit de placer le curseur sur le texte et de sélectionner l'option Colonne du menu Editer. Le texte retrouve sa forme correcte. Si vous voulez sélectionner le texte dans son intégralité : appuyez sur Control et Home pour revenir au début du fichier puis sur Shift Control End pour aller directement à la fin du fichier. Votre texte sera entièrement sélectionné.

Comment remédier à l'oubli d'une ligne vide entre deux paragraphes ?

Tout d'abord, vous devez insérer une ligne vierge entre les deux paragraphes. Ensuite, le plus simple reste à faire, puisqu'il vous suffit de "recolonner" le paragraphe fautif en sélectionnant Colonne dans le menu Editer.

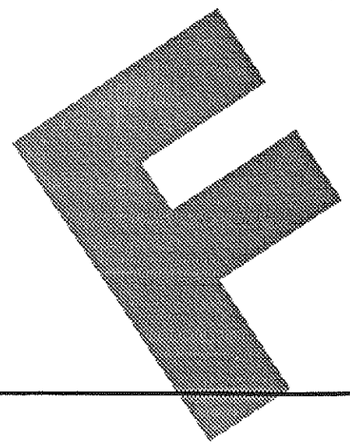
Comment se déplacer dans le texte ?

Un certain nombre de combinaisons de touches sont accessibles pour se déplacer rapidement dans le texte. En voici quelques unes :

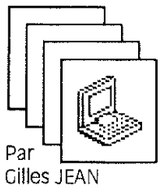
Control + Flèches verticales	:	Déplacement d'un paragraphe à un autre
Control + Flèches horizontales	:	Déplacement d'un mot à un autre
Control + Home/End	:	Déplacement au début/fin du texte
Control + Backspace	:	Efface le mot à gauche du curseur
Control + Delete	:	Efface le mot à droite du curseur
Psion + Backspace	:	Efface la partie de la ligne entre le début de celle-ci et le curseur
Psion + Delete	:	Efface la ligne, entre le curseur et la fin de la ligne
Psion + Control + Backspace/Delete	:	Efface la totalité de la ligne

Ces codes sont valides dans l'ensemble des logiciels du MC.

Ces informations vous seront de la plus grande utilité si vous utilisez beaucoup l'Editeur de textes . Alors écrivez tant que vous voulez, le MC est fait pour cela entre autre !



La Fonction Lier du MC



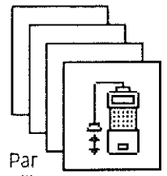
Par
Gilles JEAN

Le système d'exploitation des Mobile Computers modèles 200 et 400 a été conçu pour apporter une grande souplesse à l'utilisateur. Par exemple, le fait qu'il soit multi-tâches vous permet de faire tourner plusieurs applications en même temps. Avec ces deux modèles, les amateurs d'interfaces graphiques ont là encore été gâtés puisque le G.U.I. (Graphic User Interface) répond à toutes leurs attentes.

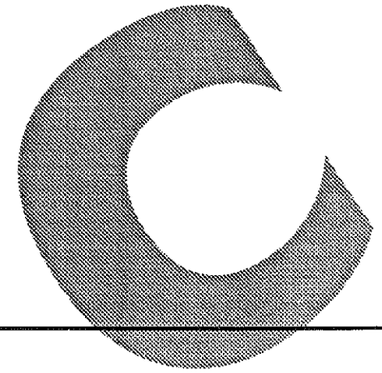
Comme sur toute interface graphique, vous disposez de fonctions permettant de faire transiter tout ou partie d'un texte vers un autre endroit de la même application. Ce sont les fameux Couper/Copier/Coller. Une option supplémentaire permet en plus au MC de passer une information d'une application vers une autre. Il s'agit de la fonction Lier.

Imaginons que vous vouliez expédier un courrier à un de vos clients enregistré dans la base de données. Ecrivez votre texte sans mentionner les coordonnées du destinataire. Lancez votre base de données et recherchez la première personne correspondant à vos critères de recherches. Sélectionnez à l'aide de votre zone tactile les informations nécessaires au bon acheminement du courrier puis cliquez à l'endroit même de votre courrier où vous voulez voir figurer l'adresse. Il vous suffit de dérouler le menu Editer et de choisir l'option Lier.

Il ne vous reste plus qu'à mettre ce principe en application avec les autres logiciels et vous vous rendrez compte de toute la puissance de cette fabuleuse machine.



Par
Gilles JEAN



Compression de Fichiers

Attention : cet article s'adresse uniquement aux personnes initiées à la programmation avancée.

La compression des enregistrements contenus dans un fichier nécessite une analyse en profondeur dudit fichier pour en améliorer les performances lors de la lecture. Les techniques à employer sont différentes et sont fonction des objectifs à atteindre.

Le gros problème de la compression d'enregistrements avec l'Organiseur est que le langage OPL ne dispose d'aucune commande ou fonction permettant d'effectuer cela simplement en compression ou en décompression. Si vous lisez un fichier de données compacté, il y a de fortes chances que celui-ci ressemble à un véritable casse-tête.

Dans l'exemple suivant, nous utilisons une technique modeste en performances mais pratique pour nos explications. En effet, cela va nous permettre de gagner environ 25% de place supplémentaire, ce qui pourrait par exemple nous permettre de faire tenir un fichier classique de 42,5 Ko sur un Datapak de 32 Ko. Intéressant, non ?

En fait, nous allons coder chaque caractère rencontré sur six bits (au lieu de huit normalement). La table ASCII sera composée de 64 caractères différents (de 0 à 63 au lieu de 0 à 255 normalement). Le caractère <TAB> est très important sur l'Organiseur. Nous lui attribuerons la valeur 60 dans notre nouvelle table.

Nous tenons à remercier tout particulièrement l'équipe du support technique PSION sans qui, ce programme n'aurait pas vu le jour.

```
eg1:
local string$(255),a%
rem Cette routine permet de compacter
rem tout les enregistrement d'un
rem fichier
open»a:main»,a,a$
do
a%=a%+1
at 1,1
print a%,»/»,count
string$=getbuff$:(1)        :rem copie l'enregistrement comme string$
string$=shrink$:(string$)   :rem compression de string$
storebuf:(string$,1)        :rem copy string$ comme enregistrement
update                      :rem mise à jour de l'enregistrement
first
until a%=count
```

```
eg2:
local string$(255)
rem Cette routine permet de compacter
rem tout les enregistrement d'un
rem fichier
open»a:main»,a,a$,b$,c$,d$,e$,f$,g$,h$,i$,j$
do
string$=getbuff$:(1)        :rem copie enregistrement comme string$
string$=grow$:(string$)     :rem decompression de string$
storebuf:(string$,1)        :rem copie string$ comme enregistrement
if disp(-1,»»)=13           :rem affichage de l'enregistrement
next                        :rem enregistrement suivant
else break                  :rem sortie de la boucle do/until
endif
until 0
```

```
getbuff$:(buff%)
local a%(3)
rem prend le contenu du buffer fichier
rem et en retourne une chaine OPL
a%(1)=$01fe
a%(2)=$2014+buff%*2
a%(3)=$3901
return usr$(addr(a%()),0)
```

```
shrink$:(string$)
rem prend la variable string$ et en
rem retourne la version compressé
local shrunk$(192),grown$(255)
local bnewlen%,length%,olda%,newa%,char%,newb%,oldchar%,,cycle%
```


Suite du programme

```
grown$=string$
length%=len(grown$)
newa%=addr(shrunk$)
olda%=addr(grown$)+1
do
char%=comp%:(peekb(olda%))
length%=length%-1
olda%=olda%+1
cycle%=cycle%+1
if cycle%>4
cycle%=1
endif
if cycle%=1
newa%=newa%+1
oldchar%=shiftl%:(char%,2)
pokeb newa%,oldchar%
elseif cycle%=2
pokeb newa%,shiftr%:(char%,4)+oldchar%
newa%=newa%+1
oldchar%=shiftl%:(char%,4)
pokeb newa%,oldchar%
elseif cycle%=3
pokeb newa%,shiftr%:(char%,2)+oldchar%
newa%=newa%+1
oldchar%=shiftl%:(char%,6)
pokeb newa%,oldchar%
else pokeb newa%,char%+oldchar%
endif
until length%=0
newb%=addr(shrunk$)
pokeb newb%,newa%-newb%
return shrunk$
```

```
grow$:(string$)
rem retourne la version decompressé de
rem la variable string$
local shrunk$(192),grown$(255)
local length%,olda%,oldchar%,char%,cycle%,volchar%
shrunk$=string$
length%=len(shrunk$)*4/3
olda%=addr(shrunk$)
do
cycle%=cycle%+1
if cycle%>4
cycle%=1
endif
if length% and cycle%<4
olda%=olda%+1
if cycle%=1
oldchar%=peekb(olda%)
char%=shiftr%:(oldchar%,2)
elseif cycle%=2
volchar%=peekb(olda%)
char%=shiftl%:(oldchar%,6)
char%=shiftr%:(char%,2)+shiftr%:(volchar%,4)
```

Suite du programme

```
else
oldchar%=peekb(olda%)
char%=shiftr%:(shiftl%:(volchar%,4),2)+shiftr%:(oldchar%,6)
endif
grown$=grown$+chr$(expa%:(char%))
elseif cycle%=4
char%=shiftr%:(shiftl%:(oldchar%,2),2)
if char%
grown$=grown$+chr$(expa%:(char%))
endif
endif
length%=length%-1
until length%=0
return grown$
```

```
storebuf:(string$,buff%)
rem lit le contenu de la variable string$
rem et la copie octet par octet dans le
rem fichier en cours
local info$(255),buffer%,oldstr%,pos%,length%,adrinfo%
info$=string$
buffer%=peekw($2014+buff%*2)
adrinfo%=addr(info$)
length%=len(info$)
do
pokeb buffer%+pos%,peekb(adrinfo%+pos%)
pos%=pos%+1
until pos%>length%
```

```
expa%:(char%)
rem prend la valeur de char% comprise
rem entre 0 et 63 et la convertie en une
rem valeur ASCII
local newchar%
if char%>=1 and char%<=59
newchar%=char%+31
elseif char%=60
newchar%=9
endif
return newchar%
```

```
shiftr%:(byte%,bits%)
rem prend byte% et la change en bits%
rem à droite de la chaine
return int(byte%/(2**bits%))
```


Par
Gilles JEAN

Encore un chronomètre...

Nous avons déjà eu l'occasion de traiter le problème du chronomètre et de l'Organiseur II dans un Bulletin Technique précédent. Aujourd'hui, nous allons expliquer par une application un peu particulière l'utilisation du compteur de trames. Ce compteur est incrémenté tous les cinquante millièmes de seconde et stocké à l'adresse $20CB. Le programme suivant, composé de trois procédures, permet de gérer un minuteur (décompte de temps) et un chronomètre. Ce programme est en fait destiné aux judokas (grands moissonneurs de médailles olympiques) afin d'effectuer le suivi d'un combat.

La première procédure permet de sélectionner, par l'intermédiaire d'un menu, la durée totale du combat :

```
JUDO:
global m,s
local m%
do
m%=menu("1Min,2Min,3Min,4Min,5Min")
if m%
m=m% :affi:
endif
until m%=0
```

La deuxième procédure permet de gérer l'affichage, les interruptions clavier et la fin d'un combat. La touche EXE permet alternativement de

commencer le décompte chrono ou d'arrêter le chrono. La touche I permet alternativement de commencer ou d'interrompre une immobilisation..

```
affi:
local k%,t1,t2,c,i1,i2,i,i%,f%
do
do
cls
print right$(gen$(100+m,3),2);":";right$(gen$(100+s,3),2)
print"EXE pour Debut" :k%=get
if k%=1
return
endif
until k%=13
m9=m :t1=peekw($20cb) :cls
do
at 1,1
print right$(gen$(100+m,3),2);":";right$(gen$(100+s,3),2),"
Immo=(";right$(gen$(100+i,3),2);")";"EXE=Stop I=Immo"
if i%
i2=peekw($20cb) :i=int((i2-i1)/20)
endif
if f%=0
t2=peekw($20cb) :c=int((t2-t1)/20) :c=(m9*60)-c
s=(60*(((c/60)-(int(c/60)))*100))/100 :m=int(c/60)
endif
k%=key
if k%=%i or k%=%I or k%=%7
i1=peekw($20cb) :i%=1-i%
elseif k%=13
do
at 1,2 :print chr$(15);"EXE=Debut"
until get=13
pokew $20cb,t2
endif
if c=0
f%=1
endif
until (c=0 and i%=0) or i=30 or k%=1
at 5,1 :print"0"
if i=30
at 14,1 :print"30"
endif
bip:(3)
m=m9 :s=0 :i=0 :i%=0 :f%=0
until 0
```

La troisième procédure permet simplement d'émettre des bips sonores à la fin du combat.

```
bip:(aa%)
local a%
do
a%=a%+1
beep 99,99
pause 5
until a%=aa%
```


Par
Gilles JEAN

L'Organiseur et les dates

Il y a déjà bien longtemps (ça commence comme un conte de fée), nous avions publié un article dans les colones de notre merveilleux Bulletin Technique (maintenant ça fait publicité) permettant de calculer l'écart numérique entre deux dates pour les Organiseur CM et XP. Un peu plus récemment, un autre article traitait la fonction DAYS qui permet de calculer le nombre de jours écoulés depuis le 1er janvier 1900.

La routine CONVDT$ permet d'effectuer le travail inverse, à savoir, le calcul d'une date à partir du nombre de jours et d'une date de référence.

Comme vous le savez déjà, l'Organiseur est une machine nécessitant parfois beaucoup d'astuces de la part des développeurs pour optimiser ou compacter les informations à stocker. Voici donc une possibilité de stocker les dates non plus sous leur forme habituelle JJ/MM/AAAA (10 octets) mais sous une forme numérique NNNNN (entre 3 et 5 octets).

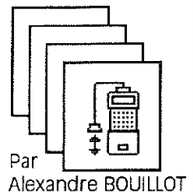
```
convdt$:(nb)
rem Module de convertion de date
rem pour des dates comprise
rem entre 1/1/1901 et 31/12/2152
local annee,mois,jour,a,t,ch$(10)
if nb>=365 and nb<=92406
a=nb+1
else raise 247
```

Suite du programme

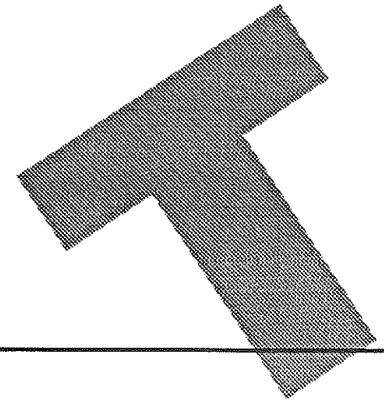
```
rem Erreur 247 = Err Argument Fnc
endif
annee=intf(a/365.25)
a=a-intf(365.25*annee) :annee=1900+annee
ch$=mid$(gen$(10000+annee,5),2,4))
rem test des années bisextiles
rem t=-1 si ok sinon t=0
if (annee/4)=intf(annee/4)
t=-1 :a=a+1
else t=0
endif
rem test mois par mois
if a>334
jour=a-334 :mois=12
elseif a>304-t
jour=a-304+t :mois=11
elseif a>273-t
jour=a-273+t :mois=10
elseif a>243-t
jour=a-243+t :mois=9
elseif a>212-t
jour=a-212+t :mois=8
elseif a>181-t
jour=a-181+t :mois=7
elseif a>151-t
jour=a-151+t :mois=6
elseif a>120-t
jour=a-120+t :mois=5
elseif a>90-t
jour=a-90+t :mois=4
elseif a>59-t
jour=a-59+t :mois=3
elseif a>31
jour=a-31 :mois=2
elseif a>0
jour=a :mois=1
endif
ch$=(mid$(gen$(100+jour,3),2,2)+" "+mid$(gen$(100+mois,3),2,2)+" "+ch$
return ch$
```

```
test:
local nb
do
cls
trap input nb
if err :stop :endif
print convdt$:(nb)
get
until 0
```

Si parmi vous, certaines personnes disposent d'algorythmes plus performants, n'hésitez pas à nous les faire parvenir par courrier et nous en ferons bénéficier tout le monde. Merci.



Par
Alexandre BOUILLOT



Transfert des fichiers notes du LZ

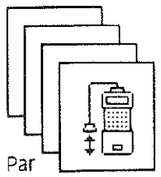
Les utilisateurs d'Organiseur LZ ou LZ 64 qui avaient un XP ou un CM avec une Comms Link en 2 lignes, nous ont souvent demmandé comment sauvegarder les fichiers Bloc notes. La mise à jour de la Comms Link n'étant pas faite, voici comment, malgré tout, réaliser vos sauvegardes.

Dans le manuel de la Comms Link, on trouve l'explication des fonctions OPL pour la gestion des communications. Deux fonctions vont nous intéresser ici ; il s'agit des fonctions qui permettent de faire des transferts avec le logiciel CL vers un Mac ou un PC (XTRECV et XTSEND). En regardant les explications, on s'aperçoit que l'on peut spécifier le type de fichier à transférer. Mais seul les types 0 à 5 sont indiqués. Si l'on sait que les fichiers Notes sont de type 7, on peut alors réaliser nos transferts. Voici deux petits programmes pour envoyer et recevoir le fichier NOTES à partir du répertoire ou du dossier courant :

```
envoi:
xtsend("notes","notes",7)
```

```
recoi:
xtrecv("notes","notes",7)
```

A vous d'étoffer ces programmes pour simplifier vos transferts et les généraliser à tous les types de fichiers.



Par
Gilles JEAN

La Gestion d'une table d'index

Pourquoi une table d'index ?

Les fichiers OPL sont tous d'accès séquentiel ; c'est-à-dire qu'à un moment donné, seul l'enregistrement qui correspond à la position courante dans le fichier est immédiatement accessible. Les fichiers OPL sont aussi de longueur variable ; c'est-à-dire que chaque élément de l'enregistrement occupe uniquement la place réelle sur le pack. Nous en déduisons logiquement que le temps d'accès à l'enregistrement sera proportionnel au nombre d'enregistrements et à la moyenne de la longueur de chaque enregistrement du fichier. La méthode que nous expliquons ci-dessous permet d'optimiser le temps d'accès à un enregistrement d'un fichier.

Il existe un nombre considérable de méthodes de gestion d'une table d'index. Elles diffèrent par la nature des composantes du fichier, les restrictions sur la taille, l'accès, les modifications et autres opérations sur les enregistrements du fichier, l'efficacité de l'interface utilisateur. En conséquence, l'OPL, comme la plupart des langages de programmation fournit quelques outils pour répondre à ce problème. L'OPL met à notre disposition des commandes permettant d'accéder à un enregistrement d'un fichier. Ces commandes sont les suivantes: FIRST, LAST, NEXT, BACK, FIND et POSITION. Nous retiendrons pour l'exemple listé ci-dessous uniquement les deux dernières commandes. Imaginons que nous disposions d'un Organiseur (peut-être même de plusieurs) pour renseigner nos clients sur la disponibilité de nos produits. L'Organiseur doit absolument nous fournir l'information le plus rapidement possible. Le programme suivant utilise la fonction FIND. Pour rappel, cette fonction permet de rechercher dans la suite du fichier la chaîne saisie dans f$ pour ramener l'enregistrement correspondant en cours. Notez que, lorsque l'enregistrement est trouvé, ce programme affiche en secondes le délai pour y accéder.

```
rech:
global f$(12),t,f%
open»a:art2",a,a$,b$
do
cls
print»RECHERCHE EXACT»
print chr$(24);
at 1,3
print»Code recherché ?»
kstat 3
trap edit f$
if err or len(f$)=0
return
else
first
pokew $20cb,0
f%=find(f$)
if f%
t=peekw($20cb)
beep 99,99
if disp(1,a.a$+chr$(9)+a.b$+chr$(9)+»Temps
réel=»+fix$(t/20,2,6))=1
break
endif
else
beep 99,99
at 1,4
print chr$(23);»NON TROUVE.»
if get=1
break
endif
endif
endif
until 0
```

Le programme suivant utilise la fonction POSITION. Pour rappel, cette fonction permet de se positionner sur l'enregistrement dont le numéro d'ordre dans le fichier est f%. Notez que, lorsque l'enregistrement est trouvé, ce programme affiche en secondes le délai pour y accéder.

```
rechpos:
global t,f%
open»a:art2",a,a$,b$
do
cls
print»POSITION EXACT»
print chr$(24);
at 1,3
print»Pos. recherché ?»
kstat 3
trap input f%
if err or f%=0
```

Suite du programme

```
return
else
first
pokew $20cb,0
position f%
if f%<=count
t=peekw($20cb)
beep 99,99
if disp(1,a.a$+chr$(9)+a.b$+chr$(9)+»Temps
réel=»+fix$(t/20,2,6))=1
break
endif
else
beep 99,99
at 1,4
print chr$(23);»NON TROUVE.»
if get=1
break
endif
endif
endif
until 0
```

Vous pouvez constater, si vous lancez ces deux programmes et si vous accédez au même enregistrement, que le délai avec la fonction POSITION est largement inférieur à celui de la fonction FIND. POSITION ne compte, en effet, que les pointeurs de fin d'enregistrement alors que la fonction FIND recherche une chaîne dans chaque enregistrement placé dans le buffer. Le principe étant acquis, il nous faut maintenant dégager une règle logique. Nous savons que la commande la plus rapide permettant de pointer sur un enregistrement est POSITION. Il faut donc faire en sorte de dégager une valeur à cette fonction avec le moins de commandes OPL possible. Pour reprendre notre exemple, l'idéal serait que le code du produit à rechercher contienne le numéro d'ordre dans le fichier Organiseur ou mieux, qu'il soit le numéro d'ordre. Le programme suivant permet de créer une table d'index dans des variables tableaux. Il va de soi que le fichier a été préalablement trié sur la variable à partir de laquelle nous faisons notre sélection.

```
index:
global p%,f%,fl%,f,t,p(500),c(500)
cls
print»CREATION DE LA TABLE»;
print»       D'INDEX»
print chr$(25);
print»Un instant»
open»a:art2",a,a$,b$
first
rem ************************************************
rem **************Chargement des variables tableaux*****
rem ************************************************
```

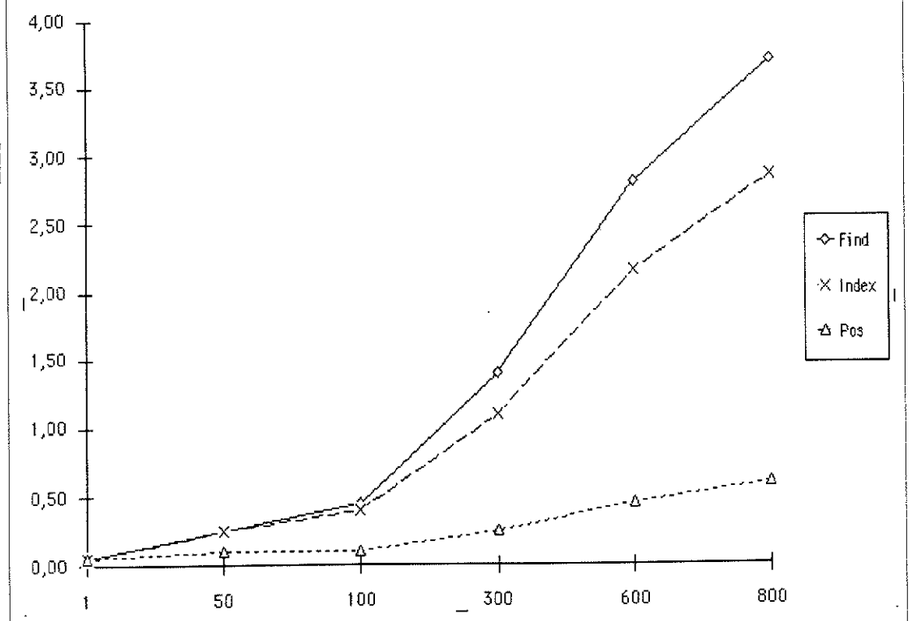


```
do
p%=p%+1
p(p%)=pos
c(p%)=val(mid$(a.a$,1,12))
at 12,4
print pos
next
until eof
f%=p%
rem ************************************************
rem **************Saisie de la recherche***************
rem ************************************************
do
cls
print»RECHERCHE EXACT»
print chr$(24);
at 1,3
print»Code recherché ?»
trap input f
if err or f<1
return
else
p%=0
pokew $20cb,0
rem ************************************************
rem **************Recherche dans les variables tableaux*
rem ************************************************
do
p%=p%+1
if f=c(p%)
fl%=1
break
endif
until p%>f%
if fl%
rem ************************************************
rem **************Acces rapide a l'enregistrement*******
rem ************************************************
position p(p%)
t=peekw($20cb)
beep 99,99
if disp(1,a.a$+chr$(9)+a.b$+chr$(9)+»Temps
réel=»+fix$(t/20,2,6))=1
break
endif
else
beep 99,99
at 1,4
print chr$(23);»NON TROUVE.»
if get=1
break
endif
endif
endif
until 0
```

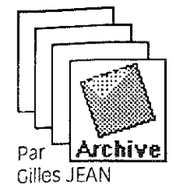
Une fois que le programme a chargé la table d'index, il vous demande de saisir le code de l'article. Le programme balaie alors la table d'index en commençant toujours par le début (variable indicée p et c avec pour indice la variable p%). Si la valeur de référence (f) est rencontrée, le programme pointe directement sur la position de l'enregistrement qui est rangé et y accède rapidement.

Le tableau ci-dessous indique les temps d'accès selon les formules (dans les programmes) citées ci-dessus pour un fichier constitué d'une rubrique de 13 caractères et d'une autre rubrique d'environ 15 caractères :

Nb enr	FIND	INDEX	POSITION
1	0.05	0.05	0.05
50	0.25	0.25	0.10
100	0.45	0.40	0.10
300	1.40	1.10	0.25
600	2.80	2.15	0.45
800	3.70	2.85	0.60



Pour conclure, il est important de rappeler que c'est l'aspect de votre fichier et sa taille qui vont déterminer la meilleure technique pour accéder le plus rapidement possible à l'enregistrement recherché. Cet article ne peut donc être considéré comme une solution à part entière, mais il espère vous amener à trouver la solution aux besoins des utilisateurs de ces merveilleuses machines que nous nommons Organiseur II.



Par
Gilles JEAN

Initiation à la programmation Archive

Archive, le générateur d'applications de PC4, permet d'envisager des réalisations touchant notre vie quotidienne tant professionnelle que familiale. L'objet de cet article n'est pas de vous apprendre à programmer sous Archive, vous comprendrez facilement que cela nécessiterait une approche particulière du problème pour être traité avec efficacité et cela n'est absolument pas l'objectif de notre Bulletin Technique.

Nous allons utilisez le fichier PAYS pour toutes les manipulations nécessaires à ces colonnes. Assurez-vous que les fichiers PAYS.DBF et PAYS.SCR figurent bien sur votre disque dur ou sur votre disquette. Si cela n'était pas le cas, recopiez ces deux fichiers à partir des disquettes originales (TUTOR pour la versions 5,25 ou EXEMPLES pour la version 3,5).

Lorsque vous démarrez Archive, vous êtes alors en mode interpréteur de clavier et vous pouvez directement tapez vos commandes qui seront interprétées après avoir été validées. Il existe toutefois un moyen permettant de construire son application ou tout simplement une partie de son application. Il suffit de créer une ou plusieurs procédures qui auront pour fonction d'effectuer le travail à votre place. Lancez Archive comme suit :

```
c:\pc4>archive
```

La zone de guidage, la zone de travail et le menu des commandes apparaissent rapidement à l'écran. Tapez la syntaxe suivante :

```
chargem «pays»
```

La commande chargem permet de charger un masque d'écran précédemment créé par la commande éditem (voir manuel page 195). Vous pourrez constater, avec un peu de pratique, tout l'intérêt de cette commande. Il nous reste maintenant à ouvrir le fichier des pays. Tapez la syntaxe suivante :

```
lis «pays»
```

Il existe deux commandes permettant d'ouvrir un fichier. La commande LIS ouvre un fichier en lecture uniquement et la commande OUVRE permet l'accès en lecture/écriture.

Nous allons maintenant effectuer les mêmes opérations en mode programmation. Tapez edite puis un nom de procédure. Afin de pouvoir démarrer directement après le chargement d'Archive, la procédure principale doit être nommée START. Nous allons, nous aussi, utiliser cette fonctionnalité pour automatiser le lancement de notre application. Tapez donc START puis return.

Notre procédure va être constituée des éléments identifiés ci-dessus. Tapez les commandes suivantes :

```
chargem «pays»
lis «pays»
ecran
```


Vous noterez que l'exécution de certaines commandes par le mode interpréteur de clavier donne un effet différent en mode programmation. Ici, la commande montre permet de forcer l'affichage des variables de l'enregistrement en cours.

Tapez ensuite deux fois sur la touche ESC du clavier afin de revenir au menu de commandes d'Archive. Il nous faut maintenant sauvegarder sur disque le contenu de notre procédure. Tapez :

```
sauve objet «pays»
```

Tapez ensuite quitte pour sortir d'Archive et revenir sous DOS. C'est maintenant que nous allons contempler notre oeuvre. Tapez sous DOS la chaîne suivante :

```
archive pays
```

Et voilà, cela fonctionne tout seul. Il ne vous reste plus désormais qu'à mettre en pratique les différents principes enoncés ci-dessus en les agrémentant à votre goût.

Dernières versions

MC

Système	1.29
Mobile TR	1.0ß
Mclink	1.29

ORGANISEUR

CM	3.3
XP	3.6
POS200	3.6
POS250	3.6
POS350	3.6
POS296	3.6
LZ	4.5
LZ64	4.5
Développ.	2.2
CL PC	2.0
CL MAC	2.0
Log. 2 lignes	
Tableur	2.0
Topfinance	2.2
Filepak	2.0
Barpak	0.9
Formulator	1.0
Travel Pak	1.5
Maths Pak	1.0
Log. 4 lignes	
Tableur	2.0
Topfinance	1.3
Filepak	1.4
Prakpak	1.0

PC4

PC4	1.1

XCHANGE

Xchange	1.37