

Database diary

Running a business is time consuming, often frustrating.

Sandra Essex shows how Archive can make life easier

COMPARED with expensive business database software, **Archive** can often be more easily used to produce very sophisticated applications. It only requires a little imagination and patience to learn the built-in procedural language, to achieve some remarkable results. The listings which follow provide an interrelated set of procedures which make up a salesman's calling diary, including associated expenses. It could easily be changed to suit individual needs, but working through the notes will show how to structure groups of procedures to obtain more elaborate programs.

The manual already shows how **Archive** commands and functions such as First, Next and Last can be entered directly from the keyboard. Using the procedures editor — type "edit", Enter — allows you to write complex procedures — procs — which can trap errors, do complicated maths functions, sort and alter files with just a few key presses. Those procs can be saved to a microdrive and loaded automatically when needed. This example assumes that a printer and monitor are attached. A TV can be used, but some adjustments to the program will have to be made to fit 64 characters on the screen. REM statements point to any **Archive1** differences.

First we create a database file — dbf — within a proc. That is good practice and avoids ending too soon by mistake. It also allows the fields to be edited before calling the proc and creating the dbf. **Proc a** in listing one shows the name of the procedure. When creating a file within a proc — **proc makefl** — the command **endcreate** must be added. The **company\$** field is arranged in ascending order before the file is closed. The use of **first,next** is then related to the field

company\$. If you want it arranged in descending order use "**order company\$d**". **Order** only works on the first eight characters, so names such as British Telecom or British Aerospace would have to be abbreviated. You can **order** up to four fields, but the more fields ordered, the fewer files **Archive** will be able to support.

Design your screen and save it as "**repcall**". Make sure you are in

entered and will later be displayed. **Archive1** allows entry outside the specified area but **Archive2** needs one extra space — dot — than the text to be entered. Limiting dots to what can be printed, say, in a mail list file, will make sure that text fits an address label. Using **sedid** makes the screen easier to read and allows more variables to be shown. Remember that the default screen is very limiting because

Listing one

```
proc a
  rem save "filename"
endproc
proc make
  create "callrep"
  company$:contact$:area$:day$:mon$:year$
  note1$:note2$:distance:due:day2$:mon2$:year2$
  vdate$:vdate2$
endcreate
order company$a
close
endproc
```

mode0 and type **sedid**. The variables display from top to bottom and left to right should read: **company\$**; **contact\$**; **area\$**; **day\$**; **mon\$**; **year\$**; **note1\$**; **note2\$**; **distance**; **due** — **due** must be set at line: 14, col: 62; — **day2\$**; **mon2\$**; **year2\$**. The **vdate\$** and **vdate2\$** do not need to be seen and so are not displayed. The dots show the screen area where the text is

you can only use **insert,alter,sinput** on fields that are displayed.

Efficient use of memory is obviously important. Whilst all procs could be saved and then loaded together, it would have the effect of seriously reducing the available working memory. Therefore, the program can be broken into three parts by saving each separately. **Proc a** shows what each

Listing two

```
proc a
  rem save "report"
endproc
proc menu
  print at 22,56; ink 4;"FILES IN RECORD "count()
  message;choice$
  print at 24,56; ink 4;"MEMORY LEFT "memory()
  let opt$="": while opt$<"e"
    let opt$=lower(getkey())
    if opt$="e"
      message;"CLOSING ALL FILES": close : mode 1,8: stop : endif
      print at 24,56;rept(" ",24)
      if opt$="i" or opt$="a" or opt$="d" or opt$="v":message;wait$
      run object "report1":rem archive(1) = run "report1"
      endif
      if opt$="p":message;wait$
      run object "report2":rem archive(1) = run "report2"
      endif
    endwhile
  endproc
proc message;mess$
  print at 23,0;rept(" ",80)
  print at 23,0; ink 2;mess$
endproc
proc start
  mode 0: let wait$="*** PLEASE WAIT FOR SYSTEM TO LOAD ***"
  let choice$="Insert (i): Alter (a): View (v): Delete (d):"
  Print out (p) End (e) ?
  print at 10,15; ink 2;wait$: open "callrep": sload "repcall"
  cls : screen :rem This line not required for Archive(1)
  menu
endproc
```

group of procedures is to be called when saved. The program is started by entering run "report". Once working the computer will drop from memory groups of procs which are not currently called.

To run a program, each group of procedures must include one called start, as that will be the first to be called. Proc start, in listing two, includes mode0 to remove the default prompts from the work area. When Archive is first loaded the mode is 1,8. The strings wait\$ and choice\$ are used throughout the program to provide a standard and present a professional image which is easy to use.

Proc menu prints choice\$ and the text is displayed, asking what action is to be undertaken. The number of files in record and remaining memory are displayed using the functions count() and memory(). Lower (getkey()) waits for a key to be pressed, then converts the letter to lower case. If 'e' is chosen the message is overwritten

Archive can be used to produce very sophisticated applications and changed to suit individual needs

with the new message while the file is being closed. If one of the other options is chosen the message;wait\$ is displayed and that whole group of procedures is cancelled from memory. Archive would then run the procs required.

Proc message;mess\$ prints any text passed to it at the print-at position on screen. Any string can be displayed, for example, message;"hello" or message; appropriate string variable.

The group of procedures in listing three is saved separately as report1. When called it goes first to proc start, which includes an error trapping routine using the command error. The while-endwhile loop in proc mistake will trap any errors in subsequent procs and return you to the menu.

In proc newrec the use of Archive's command append adds a record to the file with the information received in proc sinp. To ensure no details from an existing file are added, proc clear is called and the command sprint makes sure that the screen is ready for new

COMPANY: GREENFIELDS

DATE VISITED :- 03/06/85 CONTACT Mr Tom Jarret

NOTES

First visit. Boss is Mr Adrian Green who was absent. Jarret friendly old-timer and knows his business but Green places orders.

DATE VISITED :- 03/07/85 CONTACT Mr Adrian Green

NOTES

Requires examples of rose root stocks before ordering. Order potential considerable once Green won over.

DATE VISITED :- 03/08/85 CONTACT Mr Adrian Green

NOTES

Impressed with root stock quality. Willing to order 500 if discount improved. Can he give 20%?

DATE VISITED :- 16/09/85 CONTACT Mr Adrian Green

NOTES

Quoted 20% discount for rose root stocks but order must be 1000. Agreed will order 2000 if 25% given. Phoned Boss who agreed. To supply 11/85.

DATE VISITED :- 01/11/85 CONTACT Mr Adrian Green

NOTES

Checked what delivery date best for 2000 rose root stocks as in area. 23rd to 24th this month if we can arrange & preferably am.

AREA LONDON

FOLLOW UP VISITS FOR SEPTEMBER 1985

DATE	AREA	COMPANY	CONTACT
09/09/1985	LONDON	BAY TREE	Mr. I. White
16/09/1985	LONDON	GREENFIELDS	Mr Adrian Green

DATE	AREA	COMPANY	CONTACT
------	------	---------	---------

05/09/1985	GLOS	FLOWER GARDEN	Miss I. Glass
09/09/1985	LONDON	BAY TREE	Mr. I. White
16/09/1985	LONDON	GREENFIELDS	Mr Adrian Green
20/09/1985	SURREY	GREENFINGERS	Miss A. Broom
21/09/1985	GLOS	STALKS	Miss B. Thin
30/09/1985	SUSSEX	TREES	Mr Bill Limb

EXPENSES DUE FROM 01/01/1985 TO 31/12/1985

DATE VISITED	AREA	COMPANY	PETROL EXPENSES
01/02/1985	SURREY	GREENFINGERS	£ 3.80
03/02/1985	GLOS	FLOWER GARDEN	£ 10.60
09/02/1985	LONDON	BAY TREE	£ 1.00
14/03/1985	LONDON	BAY TREE	£ 0.80
16/03/1985	GLOS	FLOWER GARDEN	£ 10.20
18/03/1985	GLOS	STALKS	£ 9.60
27/04/1985	GLOS	FLOWER GARDEN	£ 10.80
05/05/1985	LONDON	BAY TREE	£ 0.80
06/05/1985	GLOS	STALKS	£ 9.40
11/05/1985	SURREY	GREENFINGERS	£ 3.80
03/06/1985	LONDON	GREENFIELDS	£ 3.40
10/06/1985	GLOS	FLOWER GARDEN	£ 9.40
28/06/1985	LONDON	BAY TREE	£ 0.60
03/07/1985	LONDON	GREENFIELDS	£ 3.40
07/07/1985	SURREY	GREENFINGERS	£ 4.00
09/07/1985	GLOS	STALKS	£ 9.40
03/08/1985	LONDON	GREENFIELDS	£ 3.40
04/08/1985	SUSSEX	TREES	£ 6.00
09/08/1985	SURREY	GREENFINGERS	£ 4.20
09/09/1985	LONDON	BAY TREE	£ 1.40
16/09/1985	LONDON	GREENFIELDS	£ 3.40
30/09/1985	SUSSEX	TREES	£ 6.20
01/11/1985	LONDON	GREENFIELDS	£ 3.40
			£ 119.00

input. Finally, before `append` takes place `proc check` is called. Note how `proc mistake` is used to return to the menu if necessary.

With `proc sinp`, the command `sinp` provides an alternative method of adding information. Data can be added to a displayed field in any order. That differs from `insert` in which data must be entered in the file's order of creation. In our case field, `company$` is made upper case. The dates entered are checked for errors by calling `proc getdate`. Petrol expenses are calculated when the distance is entered — let due = distance times 20 pence per mile/100 to give number of £ and p. Using the `let` command the dates are arranged in reverse order in `vdate$` and `vdate2$` so that when they are required in `proc petrol` and `proc follup` the correct period can be calculated.

`Proc getdate` checks the entries which are passed to it, taking the name `z$` and using the `Archive` functions `len` & `code`.

1st line: If only one number has been inserted a 0 is placed in front of it.

2nd line: Checks that both numbers inserted are not noughts.

3rd line: Checks that the first character is in the range 0-9.

4th line: Checks that the second character is in the range 0-9.

5th line: Checks that no more than two characters have been inserted.

If any mistakes are found by those lines, an error message is displayed — `proc mist` — and you are returned to the menu. If all is correct the value of `z$` is given to `q$`, which can then be passed back to the calling variable. For example, 2 becomes 02 but 20 remains 20.

`Proc action;act$` takes the form of any text passed to it. The text is shown on the screen and subsequent text is displayed next to it.

`Proc bye` deletes the record obtained through `proc getrec` after the required confirmation has been received.

**Data can be added
to a displayed
field in any
order and the
dates entered are
checked for errors**

Listing three

```
proc a
rem save object "report1" :rem Archive(1) = save "report1"

endproc
proc action;act$
space: print at 23,0: ink 4:act$:
endproc
proc bye
let yn$="":getrec
while yn$<>"y" or yn$<>"n"
action:"DELETE THIS RECORD : ENTER Y/N ? "
input ink 4:yn$: let yn$=lower(yn$)
if yn$="y": delete :clear: sprint : print at 22,72:
return : endif
if yn$="n":mistake: endif : endwhile
endproc
proc check
message:"THESE DETAILS O.K. TO SAVE Y/N ?"
let yn$="": while yn$<>"y" or yn$<>"n": let yn$=lower(getkey())
if yn$="n":mistake: endif : if yn$="y": return : endif :
endwhile endproc
proc clear
let company$="": let contact$="": let area$=""
let day$="": let mon$="": let year$="": let vdate$=""
let note1$="": let note2$="": let distance=0: let due=0
let day2$="": let mon2$="": let year2$="": let vdate2$=""
endproc
proc getdate; z$
if len(z$)=1: let z$="0"+z$: endif
if code(z$)<49 and code(z$(2))<49:mist: endif
if code(z$)<48 or code(z$)>57:mist: endif
if code(z$(2))<48 or code(z$(2))>57:mist: endif
if len(z$)>2:mist: endif : let q$=z$
endproc
proc getrec
action:"ENTER NAME OF COMPANY :": input coname$
let coname$=upper(coname$)
locate coname$: sprint : let show$="": while show$<>"y":pos
message:" THIS RECORD Yes. Next. Back. Fwd jump. Rev jump.
Quit. ?"
let show$=lower(getkey())
if show$="q": print at 22,15:rept(" ",20):mistake: endif
if show$="y": print at 22,15:rept(" ",20): return : endif
endwhile
endproc
proc menu
let opt$="": while opt$<>"e":message:choice$
print at 22,56: ink 2:"FILES IN RECORD "count()
print at 24,68:rept(" ",11)
print at 24,56: ink 4:"MEMORY LEFT "memory()
let opt$=lower(getkey()):option: endwhile
endproc
proc message;mess$
space: print at 23,0: ink 2:mess$:
endproc
proc mist
print at 24,0: ink 2:"MISTAKE PRESS ANY KEY TO RETURN TO MENU"
let me$=getkey(): print at 22,15:rept(" ",20)
print at 24,0:rept(" ",80):mistake
endproc
proc mistake
let n=1: while n: error menu: let n=errnum()
if n:mist: endif : endwhile
endproc
proc newrec
clear: sprint :message:"INPUT NEW DETAILS"
sinp:check: append
endproc
proc option
if opt$="e":message:"CLOSING ALL FILES"
close : mode 1,8: stop : endif
if opt$="i":newrec: endif
if opt$="a":shop: endif
if opt$="d":bye: endif
if opt$="v":view: endif
if opt$="p": print at 24,56:rept(" ",24):message:wait$
run object "report2":rem Archive(1) = run "report2"
endif
endproc
proc pos
local z
if show$="n": next : sprint : endif
if show$="b": back : sprint : endif
if show$="f":action:"JUMP FORWARD: HOW MANY RECORDS ? "
input ink 4:z: position recnum()+z: sprint : endif
if show$="r":action:"JUMP BACK: HOW MANY RECORDS ? "
input ink 4:z: if recnum()-z<0: first : sprint : else
position recnum()-z: sprint : endif : endif
if recnum()=0
print at 22,15: ink 4:"*** FIRST RECORD ***": else
if count()-1=recnum()
```

Archive1 has a serious editing fault. If you enter edit with 18 or more procs, it will crash

vdate2\$ if all the areas are to be printed but through vdate2\$ and area\$ if only one area is to be covered. Proc foll then prints out the selected files. Function month() is used to change the month from number to words.

Proc mistake. If a mistake is detected the files will be reset and reordered because the command select may have been used. All files will be reinstated before displaying the menu.

Proc notes uses the command search, starting at the first record and continuing through the file until an exact match is found for the specified field. Smith Ltd would therefore not find Smith Limited. If a match is

continued on page 30

```

print at 22,15; ink 4;"*** LAST RECORD ***"; else
print at 22,15; ink 4;"FILE NUMBER "irecnum()+1;"
endif : endif
print at 24,68;irept(" ",11); print at 24,68; ink 4;memory()
endproc
proc sinp
sinp: company$: let company$=upper(company$)
sinp: contact$,area$: let area$=upper(area$)
sinp: day$:getdate;day$: let day$=q$
sinp: mon$:getdate;mon$: let mon$=q$
sinp: year$: if len(year$)<2:mist: endif :getdate;year$
let vdate$=year$+mon$+day$
sinp: note1$,note2$,distance: let due=distance*20/100
print at 14,62;" "
print at 14,62; ink 4;due
sinp: day2$:getdate;day2$: let day2$=q$
sinp: mon2$:getdate;mon2$: let mon2$=q$
sinp: year2$: if len(year2$)<2:mist: endif :getdate;year2$
let vdate2$=year2$+mon2$+day2$
endproc
proc space
print at 23,0;irept(" ",80)
endproc
proc start
error option: let n=errnum(): if n
mist: endif :mistake
endproc
proc swap
getrec:message;"ALTER DETAILS OR PRESS ENTER":sinp:check: update
endproc
proc view
let show$="": first : sprint :pos
while show$(">")q"
message;"Next (n): Back (b): Fwd jump (f): Rev jump (r):
Quit (q) ?"
let show$=lower(getkey());pos
if show$="q": print at 22,15;irept(" ",20): endif
endwhile
endproc

```

Proc getrec uses the command locate to find the field most similar to that ordered and displays it. When previously using proc sinp, we converted the ordered field to upper case to ensure case matching.

Proc swap is used to alter existing records and we introduce the Archive command update.

Proc view displays the records for examination and calls the procedure proc pos. The latter displays the records using next, back. Also, by using the function recnum() and the command position, we are able to jump backwards or forwards.

The procedures in listing four: action;act\$ — getdate;z\$ — menu — message — mist — pos — space — view must also appear in report2 and can be moved from report1 to save typing.

Proc pout is called via proc start and proc option. It asks which print-out is required and then calls the appropriate proc.

Proc petrol prints out total expenses due between the specified dates using the command select on the field vdate\$. If any files are found, the field vdate\$ is ordered so that the printout is in chronological order. Each file found goes through the "first: while not eof()" loop, using x to add up total expenses due. The function dec — not available to Archive — is used to line up the numeric field on the right, for example, dec — due, decimal places, width. Reset restores all the files discarded when using select. As reset

invalidated the order command, we order the company\$ field and vdate\$ field.

Proc follup selects all the follow-up calls to be made in the specified month. The files are selected through

Listing four

```

proc a
rem save object "report2" :rem Archive(1) = save "report2"
endproc
proc foll
message;"PLEASE WAIT FOR PRINTOUT"
lprint tab 23;"FOLLOW UP VISITS FOR "upper(month(val(mon$)))
lprint " 19"i;e$: lprint
"CONTACT"
lprint "DATE": tab 16;"AREA": tab 32;"COMPANY": tab 62;
lprint : first : while not eof()
lprint day2$:"/"mon2$:"/"19"i;ear2$: tab 16;area$:
lprint tab 25;company$: tab 55;contact$
next : endwhile
endproc
proc follup
action;"FOLLOW UP CALLS FOR WHICH MONTH "
input ink 4;mo$:
getdate;mo$: let mo$=q$: input ink 4;" YEAR 19"i;e$
if len(e$)<2:mist: endif :getdate;e$
message;"FOLLOW UP CALLS FOR ALL AREAS Y"i"
let fw$="": while fw$(">")" or fw$(">")n"
let fw$=lower(getkey())
if fw$="n":action;"ENTER WHICH AREA TO PRINT OUT "
input ink 4;are$: endif
if fw$="." or fw$="n":message;"PLEASE WAIT WHILE FILES ARE SORTED"
if fw$="n": select area$=upper(area$) and mon2$=mo$ and
year2$=e$
if count()>0: order vdate2$a: lprint "AREA "upper
(area$);
foll: else :action;"NO CALLS FOR "
print ink 4;upper(month(val(mon$)))
print ink 4;" 19"i;e$: AREA "upper(area$);
endif : endif
if fw$="."
select mon2$=mo$ and year2$=e$
if count()>0: order vdate2$a:foll: else
action;"NO CALLS FOR "
print ink 4;upper(month(val(mon$)))i" 19"i;e$;
endif : endif
if count()=0: print ink 4;" : PRESS ANY KEY FOR MENU"
let no$=getkey(): endif
reset : order company$a,vdate$a: return
endif : endwhile

```


Listing four continued

```

endproc
proc mistake
  let n=1: while n: error menu: let n=errnum()
  if n: reset: order company$ia,vdate$ia:
  mist: endif: endwhile
endproc
proc note
  action:"ENTER NAME OF COMPANY:"
  input coname$: let coname$=upper(coname$)
  search company$=coname$: if found()
  message:"PLEASE WAIT FOR PRINT OUT"
  lprint "COMPANY: "icompan$
  lprint rept("=",10+len(company$)): lprint
  while found()
    lprint "DATE VISITED: "ida$/"mon$"/"year$
    lprint tab 30:"CONTACT "icontact$
    lprint "NOTES: "lprint notel$
    lprint note2$: lprint: continue: endwhile
  else: action:"NO COMPANY CALLED"
  print ink 4:coname$: "PRESS ANY KEY FOR MENU"
  let no$=getke(): endif
endproc
proc option
  if opt$="e": message:"CLOSING ALL FILES"
  close: mode 1,8: stop: endif
  if opt$="l" or opt$="a" or opt$="d"
  print at 24,56: rept(" ",24): message: wait$
  run object "report1": rem Archive(1) = run "report1"
  endif
  if opt$="p": pout: endif
  if opt$="v": view: endif
endproc
proc petrol
  local x: let x=0
  action:"EXPENSES FROM DATE: ENTER DAY"
  input ink 4:da$:
  getdate:da$: let da$=q$: input ink 4:"ENTER MONTH "imo$:
  getdate:mo$: let mo$=q$: input ink 4:"ENTER YEAR 19"ie2$:
  if len(ie2$)<2:mist: endif: getdate:ie2$: let expen$=ie2$+mo$+da$
  action:"EXPENSES TO WHICH DATE: ENTER DAY"
  input ink 4:da2$:
  getdate:da2$: let da2$=q$: input ink 4:"ENTER MONTH "imo2$:
  getdate:mo2$: let mo2$=q$: input ink 4:"ENTER YEAR 19"ie2$:
  ie2$+mo2$+da2$
  if len(ie2$)<2:mist: endif: getdate:ie2$: let exp$=
  message:"PLEASE WAIT WHILE FILES ARE SORTED"
  select vdate$>expen$ and vdate$<exp$
  if count()>0: order vdate$ia
  message:"PLEASE WAIT FOR PRINT OUT"
  lprint tab 20:"EXPENSES DUE FROM "ida$/"mon$"/"19"ie2$:
  TO:
  lprint da2$/"imo2$"/"19"ie2$: lprint
  lprint "DATE VISITED: "tab 15:"AREA: "tab 32:"COMPAN: "
  lprint tab 64:"PETROL EXPENSES: "lprint
  first: while not eof()
    lprint day$/"mon$"/"19"ie2$: tab 16:area$:
    lprint tab 25:company$:
    lprint tab 68:"f"idec(due,2,8)
    rem Archive(1) = lprint tab 68:"f" idue
    let x=x+due: next: endwhile
    lprint tab 68:"=====
    lprint tab 68:"f"idec(x,2,8)
    rem Archive(1) = lprint tab 68:"f" ix
    lprint tab 68:"=====": else
    action:"NO PETROL EXPENSES DUE FOR THAT PERIOD"
    print ink 4:"PRESS ANY KEY FOR MENU: let no$=getke():
    endif
  reset: order company$ia,vdate$ia
endproc
proc pout
  message:"Petrol expenses (p): Follow up dates (f): Notes (n)
  "
  let pfc$="": while pfc$<>"p" or pfc$<>"f" or pfc$<>"n"
  let pfc$=lower(getke())
  if pfc$="p": petrol: return: endif
  if pfc$="f": follup: return: endif
  if pfc$="n": note: return: endif: endwhile
endproc
proc start
  error option: let n=errnum(): if n
  reset: order company$ia,vdate$ia:mist: endif: mistake
endproc

```

Archive

Archive2 has many extras that makes it a must for serious users, and has more memory than Archive1

found the "while found() continue endwhile" loop will print out details of all the matching records.

If you break into the program, entering: **mode0: screen: mistake** will return you to the menu.

To make editing easier, try not to make your proc lines any wider than the screen. Never use **lprint** if a printer is not attached as **Archive** will then crash.

While must always be matched with endwhile; all with endall; if with endif. Each use of while, if or all moves a procedure two spaces to the right. The indentation stops when an endwhile, endif or endall command is added, and at the end of the procedure the endproc must be two spaces from the dividing line.

Archive1 has a serious editing fault in that, when no work area is displayed — mode 0 — and you enter edit with more than 18 procs, it will crash.

It is good policy to produce backup copies of all files on a separate cartridge. Do that by replacing the **Archive** cartridge and entering backup "filename.dbf" as "mdv1_filename.dbf". As **Archive2** will not overwrite the same name, you must kill the file in mdv1 first.

Archive2 has many extras that makes it a must for serious users. The screen editor is more versatile and two more lines are available for the screen display. The variables can be coloured using the **ink** and **paper** commands. **Archive1** procs can be run on **Archive2** but the designed screens are incompatible.

In **Archive2** the screen display determines the number of characters that can be inserted. Scrolling with **Archive1** is considerably slower than with **Archive2**. Procedures can be Saved, Run, Merged and Loaded with the option **protect**, preventing anyone from listing your procs. Files saved with the option **object** — as used in this program — are given the file extension **pro** which makes loading faster. Many useful new functions have also been added.