

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	PDT
DECnet	Digital Logo	RSTS
DECsystem-10	EduSystem	RSX
DECSYSTEM-20	IAS	UNIBUS
DECUS	MASSBUS	VAX
DECwriter	PDP	VMS
		VT

UNIX is a trademark of Bell Laboratories.

Copyright © Digital Equipment Corporation 1984
All Rights Reserved

It is assumed that all articles submitted to the editor of this newsletter are with the authors' permission to publish in any DECUS publication. The articles are the responsibility of the authors and, therefore, DECUS, Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in the document. The views herein expressed are those of the authors and do not necessarily express the views of DECUS or Digital Equipment Corporation.

GOOD NOTATION IS A GOOD TEACHER

Lee Wilcox, Department of Physics
SUNY Stony Brook N.Y. 11791

I don't ordinarily preach to the converted, but perhaps some of your readers are experiencing the frustration I experience in dealing with school administrators and colleagues who are totally ignorant of APL. If these words can help your readers in discussions with such persons, I'd be pleased.

APL is a graceful notation suited for application to general education from fourth grade on. No current computer language comes close to this in affording simplified machine interaction while enriching human thought.

What is APL?

APL is both a notation and a programming language. The embryonic notation, published in 1962, by Kenneth Iverson brought clarification and consistency to diverse notations then used in applied mathematics. Several years passed before Iverson and Falkoff led a team to implement APL (A Programming Language) at IBM Corporation.

APL is interactive, terse, and powerful. In commerce there's a high demand (relative to supply) for APL programmers; pay is high because programmers wielding APL power can be so productive. APL is best known in Europe and Canada.

In U.S. schools and colleges APL has been little noticed amid the BASIC din. To appreciate APL one needs an APL terminal. But, to purchase an APL terminal (about \$200 extra) educators needed to appreciate APL: CATCH-22.

New Concepts

Come all ye Presidents, Deans, Principals, School Boards; sit at my APL terminal. Let me show why APL is uniquely suited to needs in general education. Let's talk about computing as a vehicle for IDEAS. At an APL terminal I might type:

```
DEFINE 'AVERAGE' { (+/⍵)÷#⍵ }
⍵←A←AVERAGE ⍵←DATA←?15#9
4 7 4 9 2 2 2 3 7 9 9 1 9 4 9
5.4
)SAVE
```

Lines not indented are output from the computer. DEFINE is Iverson's famous "forming function" which converts function specifications into machine programs. The symbol ⍵ is a place-saver, a stand-in for any argument one might pass to the function: AVERAGE. The variables of APL are lists or strings or rectangular arrays of arbitrarily many dimensions. ⍵←X is a command to display X. The computation proceeds from right to left so that DATA is displayed first.

In only three lines I specified a procedure named: AVERAGE, ran this procedure with an argument, assigned a variable (A) to be the result and displayed its value. Finally, I saved both the variable and the procedure in my private library. Gone are: DIMENSION, PRINT, READ, DO, GOSUB, NEXT, LOAD, LINK, MAP... APL eliminates all this. Memory allocation is automatic, type declarations are unnecessary.

Many APL concepts are new, not only to automatic computation but new also to the way in which computation is conceived.

Primitive Functions

Symbols such as: + and × denote functions that apply to arguments at left and right and return results. Learning to use a computer means: learning the meanings of such "primitive functions" and learning to combine these to produce new functions suited to tasks required for a particular subject.

A well-chosen set of primitive functions gives APL its expressiveness. Like international road signs, key-strokes representing primitives have been designed to suggest what they mean—e.g.

⍋(REVERSE) ⍋(SORT-UP) ⍋(TAKE) ⍋(DROP)

```
⍵←4↑⍵←⍋⍵←DATA[⍋⍵←DATA]
4 7 4 9 2 2 2 3 7 9 9 1 9 4 9
1 2 2 3 4 4 4 7 7 9 9 9 9 9
9 9 9 9 9 7 7 4 4 4 3 2 2 2 1
9 9 9 9
```

Primitives, like those above, approximate the logic repertoire of a first-grader—e.g. a child can put numbered blocks in sorted order before he can add. The basic meanings of a dozen or so APL primitives can be learned in a few hours. These tokens are useful in ordinary conversation. I prefer: A←⍋B to an awkward: "A is B with its components taken in reverse order". I prefer teachers who express simple ideas simply and suggestively.

[continued on page 3]

NOTABLE AND QUOTABLE

John Backus
Leader of the original FORTRAN development team

(Commenting on current trends in FORTRAN extensions to an ACM SIGPLAN conference in June, 1978)

I'm not in favor of any of them. I think conventional languages are for the birds. They're really low level languages. They're just extensions of the von Neumann computer, and they keep our noses pressed in the dirt of dealing with individual words and computing addresses, and doing all kinds of silly things like that, things that we've picked up from programming computers; we've built them into programming languages; we've built them into FORTRAN; we've built them into PL/I; we've built them into almost every language. The only languages that broke free from that are LISP and APL, and in my opinion they haven't gone far enough. [from R. L. Wexelblat, ed., History of Programming Languages, New York, Academic Press, 1981. Thanks to Homer Hartung for bringing this to the Editor's attention.]

Copyright(c)1984, Digital Equipment Corporation.
All rights reserved.

It is assumed that all articles submitted to the editor of The Special Character Set or his representatives are with the author's permission to publish in any DECUS publication and that the author has the right to grant such permission. These articles are the responsibility of the authors and, therefore, DECUS, the APL Special Interest Group, the Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in The Special Character Set. The views expressed are those of the authors and do not necessarily express the views of DECUS, the APL Special Interest Group, the Digital Equipment Corporation or the editor.

The Special Character Set is the official publication of the APL Special Interest Group of the Digital Equipment Computer Users Society. Unsolicited manuscripts are welcome.

For subscription information and an application, contact:

Membership Services
DECUS
249 Northboro Road BP02
Marlboro, MA 01752

APL SIG STEERING COMMITTEE

Chairman: Larry LeBlanc
Teletype Corp.
2330 Eastern Ave.
Elk Grove, IL 60007
312-860-8181

Library
Coordinator: Susan Abercrombie
Ventrex Laboratories
217 Read Street
Portland, ME 04103
207-773-7231

Standards
Representative: OPEN

Newsletter
Editor: Douglas Bohrer
Bohrer and Company
903 Ridge Road Suite 3
Wilmette, IL 60091
312-251-9449

Symposia
Coordinator: OPEN

European
Contact: Jean-Pierre Barasz
BARTS
54 Rue d'Amsterdam
75008 Paris, France
33-1-633-3297

STYLE SHEET

by Douglas Bohrer
Bohrer and Co.
903 Ridge Rd Suite 3
Wilmette, IL 60091

We look forward to your contribution to The Special Character Set. Your cooperation on a few matters of style would help get the newsletter out on time. The newsletter staff is all volunteer and the time you save us is appreciated.

Set-Up

Basically, your contribution should look like this article. Use black ink. Type your text single spaced. Use only one side of each sheet of paper. Skip one line between paragraphs. Set your margin at 4 3/8 inches wide. For type with 10 characters per inch, you would have 43 characters per line. For type with 12 characters per inch, you would have 52 characters.

Titles

Please start the title of your article on the left margin using all capital letters. Skip one line, then put your name, title, firm and mailing address. (You may omit your title, firm and mailing address for reasons of modesty, privacy, shame or whatever.) Sub-heads in your article should begin at the left margin with a blank line above and below them.

Letterhead will not be reproduced. Drawings should be either 4 3/8 or 9 inches wide.

Commercial Guidelines

The newsletter will follow strictly the guidelines related to the non-commercial nature of DECUS. If you have any questions on these, please contact the newsletter editor or the DECUS office.

CONTENTS

GOOD NOTATION IS A GOOD TEACHER.....	1
NOTABLE AND QUOTABLE.....	1
AND THEN THERE WERE 3.....	3
NEW DEC APL PRODUCT MANAGER.....	3
APL AT ANAHEIM.....	3
A PLOTTING WORKSPACE.....	6
LARGE DATABASE HANDLING WITH APL-NET..	13
ADVERTISING UNLIKELY.....	16
DECUS EUROPE LETTER.....	18
US SUBSCRIPTION REPORT.....	19
DECUS AUSTRALIA LETTER.....	20
SURVEY RESULTS.....	21
DEAR APL FANATIC.....	22

SUITE TERMINAL

The terminal for our hospitality suite in Anaheim will be an LA12-AB with APL keycaps provided by Bruce Hunter (617-493-4963), DEC's marketing wizzard for the Correspondent. The LA12 comes standard with an APL character set. APL keycaps are slightly extra.

AND THEN THERE WERE 3

Douglas Bohrer
Editor, The Special Character Set

Kevin Walker, formerly an APL SIG Steering Committee member, has accepted a position with Digital Equipment Corporation. As a result, he has resigned his Steering Committee position. We are down to 3 Steering Committee members. We are getting dangerously close to a forced merger with another SIG. If you plan to attend a U.S. Symposium in the next year you could become a Steering Committee member, with emergency powers for life. The current Steering Committee is prepared to be very flexible on your duties. We will need someone to represent the APL SIG at various committee meetings at the Symposium. If you can help, please contact Doug Bohrer at 312-251-9449, Sue Abercrombie at 800-341-0463 or Larry LeBlanc at 312-860-8181.

NEW DEC APL PRODUCT MANAGER

Douglas Bohrer
Editor, The Special Character Set

Dave Quigley has been named APL Product Manager for Digital. He replaces Roger Matus, former DECUS APL SIG Chairmen. Roger has gone on to other duties within DEC. Dave promises enhanced cooperation with the SIG. He says that a lot of engineering effort is going into VAX-11 APL enhancement. Dave refused to confirm or deny rumors that STSC is working on an APL for the Rainbow. He did say that he would be happy to help with questions on DEC's APL products for the VAX and DEC-10/20. His address is ZK2-3/Q08, 110 Spit Brook Rd., Nashua, NH 03062, phone 603-881-2343. Turning to another Digital development, Greg Adams, a DEC RT-11 guru, mentioned the "possibility" that APL would be one of the "national character sets" implemented on the PRO under RT-11 while answering questions at the DECUS EUROPE Symposium in Amsterdam. If it happens, your editor will try to be a field test site.

APL AT ANAHEIM

Susan Abercrombie
Ventrex Laboratories
217 Read Street
Portland, Maine 04103

At Anaheim, the APL SIG is still small, but still high quality. We are repeating two sessions which played to enthusiastic audiences in Cincinnati. An introduction to APL will highlight the reasons for using APL and a discussion of the applications for which APL is especially well suited, together with a brief tutorial. A panel on the (nearly) free APL interpreters available through DECUS (APL-11 for RSTS, RSX, and RT; SCI-APL for VAX) will describe these products and give users a chance to discuss them with experts.

DIGITAL (remember them?) has scheduled a session to discuss the features of VAX-11 APL. DEC's presentations on this product have been excellent in the past, and we haven't had a DEC session for several symposia, so this should be of particular interest.

We will try to have an APL terminal available to allow us to give hands-on demonstrations using one of the exhibit hall VAX systems.

California or bust!

GOOD NOTATION [continued from page 1]

Array Thinkings

While LISP and LOGO encourage users to think in terms of hierarchical list structures, APL encourages users to think in terms of rectangular arrays. Such data structures commonly arise in physics, engineering, social science... Observe, please, how easily one may manipulate arrays with APL primitives:

	15	(Index vector)
1 2 3 4 5		
	⍴←TABLE+(15)*,x(15)	(outer product)
1 2 3 4 5		(times table)
2 4 6 8 10		
3 6 9 12 15		
4 8 12 16 20		
5 10 15 20 25		
	+/TABLE	(sum along rows)
15 30 45 60 75		
	x/TABLE	(column products))
120 3840 29160 122880 375000		
	1 ⍴TABLE	(main diagonal)
1 4 9 16 25		
	^/,TABLE=TABLE	(is TABLE symmetric?)
1		(yes)
	⍩TABLE	(matrix reciprocal)
DOMAIN ERROR		(aha! determinant=0)

Sylvester, renowned teacher and father of matrix algebra, would surely find existential joy in APL.

Operators and Insight

Most iterative loops which one would need to write in FORTRAN, BASIC, or PASCAL are eliminated by APL OPERATORS e.g.

/(REDUCTION) \ (SCAN) *. (EACH-WITH)

(EACH-WITH) was used above to create a multiplication table. To sum a list of numbers simply write: +/LIST. A loop is implicit. It is as if the function +/1 2 3 4 is evaluated recursively as follows:

+/1 2 3 4	(question)
(+/1 2 3)+4	(bes the question)
((+/1 2)+3)+4	(again)
((3)+3)+4	(question trivial)
(6)+4	(climb out)
10	(result)

APL allows recursive definition i.e. function specifications may entail the function being specified (BASIC and FORTRAN do not). An explicitly recursive function equivalent to +/- could be designed as follows:
 A trivial case occurs when only two numbers are to be added:

```
DEFINE 'ADD:W[1] W[2]'
```

Then the general case is handled by:

```
DEFINE 'SUM:(SUM T1+W) T1+W:2=F W:ADD W'
```

This is an IF-THEN-ELSE construction which says, almost literally, SUM all but the last number, + that to the last number, T1+W UNLESS only two numbers remain, 2=F W: THEN (in that case) the result is ADD W. This suspenseful mode of reasoning may seem strange to a neophyte but RECURSIVE THINKING is a powerful paradigm, well worth practicing. I call this "the art of begging the question" Douglas Hofstadter spends much of his book (Godel, Escher and Bach) extolling this art. So often recursive functions provide easy and clear solutions. Recursion is absolutely essential in any notation suited to general education.

The dyadic primitive, L(FLOOR) returns the smaller of its two scalar arguments. Substitute L in place of + in the above example. Then you will see that L/LIST returns the least member of LIST so that L/ is (MINIMUM-OF), x/ is (PRODUCT-OF) and so on. While FUNCTIONS change data into other data, OPERATORS change functions into other functions as in: Sin x=-Cos x'. The ' (derivative) acts upon the function to its LEFT to change it into another function. The symbol / represents an OPERATOR analogous to ' . The family of functions generated by f/ (where f is any primitive function) are all are constructed on an identical recursive template. I had never noticed a kinship between SUM-OF and MINIMUM-OF nor had I appreciated any distinction between operator and function. Gifted teachers dispense pearls of insight.

If B is any Boolean vector (1s and 0s) Then deMorgan's theorem is expressed:

$$(\wedge/B) = \vee(\wedge/B)$$

(\wedge means AND OR NOT) Before I saw this elegant expression, I had trouble remembering deMorgan's theorem. Now I could never forget it! Good teachers find vivid ways to express important ideas.

Brevity and Wit

To further demonstrate the nimble brevity of APL consider:

```
DEFINE 'INDEX:( $\wedge$ (1/W)@W $\wedge$ =a)11'
```

This useful function returns the starting index of the right hand string in the left hand string.

```
W←'PETER PIPER PICKED' INDEX 'PICKED'
```

This function uses \wedge (each-with comparison) and 1 (index-in) -e.g.

```
'PETER' 1 'T'
```

3
 The design of INDEX is perhaps not obvious at a glance but when the algorithm has once been understood, (10 minutes reflection), one has acquired insight which is immediately transferable, INDEX becomes an idiom of the language. To produce the program INDEX without APL's elegant OPERATORS, one needs to construct a loop within a loop (See Niklaus Wirth, Scientific American, Sept 84)

Here is an utterly trivial function to sum the odd terms in a list:

```
DEFINE 'SUMODD:+(2|W)/W'
```

How does this work? Let's find out. Turn on the OBSERVE feature.

```

)OBSERVE
SUMODD ?10?9
SUMODD ?10?9
^
9 9 9 9 9 9 9 9 9 9
SUMODD ?10?9
^
9 8 2 9 8 4 8 6 2 7
Z++/(2|Y)/Y
^
SUMODD [1] 1 0 0 1 0 0 0 0 0 1
+/(2|Y)/Y
^
SUMODD [1] 9 9 7
Z++/(2|Y)/Y
^
SUMODD [1] 25
25

```

The OBSERVE feature reveals each minute step in a computation. SUMODD uses 2| (modulo-2) to locate odd terms then uses / (LOGIC-COMPRESS) to de-select even terms. The operator / is used in two guises as (logical-compression) and as (sum-reduction). Please compare this expression, SUMODD, with corresponding programs in PASCAL, BASIC (16 lines), FOURTH (8 lines) LISP (6 lines) given by Lawrence Tesler, Scientific American September 84. How can we admire 16 lines of BASIC for such an utterly trivial function? The very concept of a function which takes an argument and returns a value is absent in BASIC. If brevity is the soul of wit, BASIC is dimwitted indeed. I'm attracted by teachers with a trenchant wit.

In Classroom and Laboratory

Had I a class already prepared for APL by no more than three class periods, I could write on the blackboard for note-takers:

```
POLY:(a*x+1/a)+.x a
```

while explaining that this is the polynomial function with coefficients given by a. I could demonstrate:

```
Q←1 1 1 POLY Q←x+1 2 3 4
1 2 3 4
3 7 15 21
```

There is a triple benefit from such comportment. 1) I provide an unambiguous definition. 2) I teach a worthwhile notation while using it in context. 3) I show how to automate a computation. NO OTHER COMPUTER LANGUAGE CAN BE USED IN THIS WAY. APL can be woven into the warp of a subject. BASIC, FORTRAN and PASCAL can not! A fabric woven with hemp wool and silk warp could have small appeal. Iverson's direct definition notation is a "meta-APL" which suppresses computer details and abets clear, uncluttered exposition.

"APL is difficult"

To imagine that APL is difficult because it uses a dozen or so exotic new symbols is to be misguided. One might as well conclude that the difficulty of mathematics stems from its use of the Greek alphabet. APL, like mathematics, is difficult because thinking is difficult. We need all the help we can get from intelligent notation. APL primitives are metaphors worth learning and sharing, quite apart from their use in automatic computation.

If you don't use international SYMBOLS, you will need provincial RESERVED-WORDS. In Apple-soft BASIC, for example, I can't name a variable: DATA, it has to be: DATA%. But that won't work either because this would be interpreted: D AT A% because AT is a RESERVED-WORD and Apple-soft ignores blanks! What is worthwhile about this? This is an unwelcome kind of difficulty; having nothing to do with ideas.

In APL one may name variables and functions however one pleases-e.s.

```
SLOPE←FIJ CAPITAL AND CORN
```

I use underlines, for readability, to distinguish functions from variables at a glance but the language does not require this. If the task at hand involves only + and x then other primitive functions simply don't exist. It is still true that:
2+2x2

6

Consequently one can use APL from day 1 (realistically, at college level, after 5 or 10 hours reading and practice at a terminal). In this sense APL is by far the easiest extant language to use. On the other hand, because the notation is so rich and powerful, it is difficult to master. That makes it interesting.

The Future of APL in Education

In some of our finest Universities APL has flourished for years in undergraduate instruction and in sophisticated research (e.s. Harvard, Yale, UC Berkeley).

Now APL is flourishing in personal computers. High cost which inhibited APL use in academe has all but vanished. Already there are four APL interpreters to run on big blue's PC. Other versions will run on Motorola 68,000 micro machines (e.s. Macintosh, Amere) The Amere machine is a marvel of advanced technology (Jean) it has APL in ROM, 256 kbytes of RAM and a miniature disk (400 kbyte). This, lap size, machine will retail for about \$1500 (1984). We will see APL integrated with vivid graphics on the Macintosh and other PCs. A decision to introduce APL massively into general education is feasible NOW.

So we are back to CATCH-22. Powers with the purse in Education have never seen APL. Not one professor in fifty, I suspect, has any idea how computing might be incorporated into class offerings (I exclude computer science of course). Many teachers feel a responsibility to do this but don't know how to begin. Others consider computers irrelevant or worse. The way IS difficult. Computing can't simply be flunked onto an already overburdened curriculum. You don't make a successful motion picture by photographing the Progenium arch from a camera cast in cement. You don't simply translate text book equations into a computer language; that does not work! That does not weave a presentable fabric. New weavers will write new texts.

HELP!

DEC company is aggressively marketing PRO computers to colleges. My campus just contracted for 200 of these with (you guessed it) BASIC, FORTRAN, and PASCAL! From what I can learn, DEC company has no plans to develop an APL interpreter for the PRO. What a pity! Can someone out there in DECUSLAND suggest a respectable APL to run on the PRO under DOS? Where might one purchase a suitable character ROM for the PRO?

A Plotting Workspace

Paul Kobrin
American Petroleum Institute
Washington, D.C. 20005

This article describes the contents of a plotting workspace we use under VAX-11 APL. The functions in it are capable of turning a numeric vector or matrix into a scattergram on a character terminal.

The plotting function is called, simply enough, PLOT. It takes the vector or matrix to be plotted as its argument and returns an explicit result, a character array containing the graph. PLOT may be called from other functions or executed directly. When the epsilon symbol in it is converted to the more common execute primitive, it will run under non-DEC APL's.

Depending on how certain options are set, PLOT will produce a variety of graphs. If its argument is a vector, the elements are plotted as vertical values against the integers as the horizontal values. Similarly, if the argument is a matrix, each column is plotted separately against the integers. To the extent possible, a different print character is used for each column. Alternatively, if the numeric scalar .ZCOL exists, its value specifies the number of the column to contribute the horizontal values. For example, column 1 might contain x values while columns 2 and 3 contain two corresponding sets of y values.

The options are set by creating and initializing global variables all of whose names begin with an underlined character. .ZCOL, previously mentioned, is one such variable. Other global variables permit assignment of a title and axis labels, logarithmic scaling of either or both axes, choice of plotting characters and control of horizontal and vertical size of the graph.

In addition to the PLOT function and the five functions which it invokes, the workspace contains another function called EZPLOT which may be used to run PLOT. EZPLOT is niladic and does not return an explicit result. Its purpose is to provide a conversational, menu-based method of setting the global variables just described. EZPLOT contains a help capability which displays the character array EZPLOTHOW. Finally, the workspace contains a character array called PLOTHOW which describes the PLOT function.

The ^{seven}~~six~~ functions and two HOW variables in the workspace are displayed below. Following that display is a sample session using EZPLOT to generate a scattergram of one column of random numbers plotted against another column of random numbers. Note that the data vector X is created before EZPLOT is run and is specified within EZPLOT.

PLOTHOW

PLOT IS A MONADIC FUNCTION WHICH RETURNS AN EXPLICIT RESULT, A CHARACTER MATRIX WHICH CONTAINS A GRAPH. THE FULL FUNCTIONALITY OF PLOT IS AVAILABLE THROUGH THE CONVERSATIONAL INTERFACE FUNCTION EZPLOT. THE FOLLOWING DOCUMENTATION IS PROVIDED TO USE PLOT DIRECTLY, PERHAPS AS PART OF A LARGER APPLICATION.

THE ARGUMENT OF PLOT IS A NUMERIC VECTOR OR MATRIX. IF A VECTOR, ITS ELEMENTS ARE PLOTTED ON THE Y AXIS WITH INTEGERS ON THE X AXIS. IF A MATRIX, THE ELEMENTS OF EACH COLUMN ARE PLOTTED ON THE Y AXIS WITH INTEGERS ON THE X AXIS IF THE SCALAR .ZCOL DOES NOT EXIST. HENCE, THERE WILL BE AS MANY CURVES AS THERE ARE COLUMNS IN THE MATRIX. IF .ZCOL IS ASSIGNED A NUMBER, THE COLUMN WITH THAT NUMBER IS PLOTTED ON THE X AXIS.

IF THE CHARACTER VECTOR .ZTITLE EXISTS, ITS VALUE WILL APPEAR AS A TITLE CENTERED ABOVE THE GRAPH. IF THE CHARACTER VECTOR .ZXLABEL EXISTS, ITS VALUE WILL APPEAR AS AN AXIS BELOW THE X AXIS. IF THE CHARACTER VECTOR .ZYLABEL EXISTS, ITS VALUE WILL APPEAR VERTICALLY NEXT TO THE Y AXIS.

IF THE VARIABLE .ZYLOG EXISTS (ANY VALUE), THE Y AXIS WILL HAVE A LOGARITHMIC SCALE. SIMILARLY, IF .ZXLOG EXISTS, THE X AXIS WILL HAVE A LOGARITHMIC SCALE.

IF THE CHARACTER SCALAR OR VECTOR .ZCHARS EXISTS, EACH CURVE WILL BE ASSIGNED ONE OF ITS CHARACTERS AS A PLOT CHARACTER. THE FIRST CURVE GETS THE FIRST CHARACTER, ETC. IF THERE ARE MORE CURVES THAN CHARACTERS, THE CYCLE OF CHARACTERS IS REPEATED. DEFAULT CHARACTERS ARE PROVIDED IF THIS VARIABLE DOES NOT EXIST.

EZPLOTHOW

EZPLOT IS A CONVERSATIONAL FRONT END USED TO SET THE PARAMETERS OF PLOT, A FUNCTION WHICH GRAPHS ONE OR MORE CURVES ON A PAIR OF AXES. EZPLOT IS RUN BY ENTERING ITS NAME.

A MENU APPEARS OFFERING FOUR CHOICES. THE FIRST IS TO SET, ALTER OR DISPLAY THE VALUES OF THE GRAPHIC PARAMETERS. THIS MUST BE EXECUTED BEFORE A GRAPH CAN BE PLOTTED. THE SECOND CHOICE PLOTS THE GRAPH. THE THIRD CHOICE BRINGS UP THIS HELP TEXT. THE FOURTH CHOICE EXITS THE FUNCTION. THE REST OF THE TEXT DEALS WITH CHOICE 1.

ONCE CHOICE 1 IS SELECTED, DESCRIPTIONS OF THE 11 PARAMETERS ARE DISPLAYED ALONG WITH THEIR DEFAULT AND CURRENT VALUES. THE USER IS PROMPTED TO RETURN TO THE MENU BY TYPING 0 (ZERO) OR TO ENTER THE NUMBER OF THE PARAMETER TO BE ALTERED. IF AN INTEGER FROM 1 TO 11 IS ENTERED, THE USER WILL BE PROMPTED AGAIN, THIS TIME TO RETURN TO THE MENU BY TYPING M, TO RESTORE THE DEFAULT OF THE CHOSEN PARAMETER BY TYPING D, OR TO SET A VALUE BY TYPING IT. DO NOT ENTER A VALUE THAT LOOKS LIKE THE DEFAULT; TO GET THE DEFAULT, ENTER D. THE FIRST SEVEN PARAMETERS ARE FULLY DESCRIBED WHEN THEY ARE DISPLAYED. HENCE, ONLY THE REST WILL BE DISCUSSED BELOW. NOTE THAT ONLY PARAMETER 10 NEEDS TO BE SET; DEFAULTS SUFFICE FOR THE OTHERS.

PARAMETER 8 SELECTS PRINT CHARACTERS. THE FIRST PRINT CHARACTER IS FOR THE FIRST CURVE, ETC. CHARACTERS WILL BE USED IN CYCLE IF THERE ARE MORE CURVES THAN CHARACTERS.

PARAMETER 9 SELECTS THE COLUMN OF THE INPUT MATRIX WHICH WILL BE PLOTTED ON THE X AXIS. OTHER COLUMNS WILL BE PLOTTED ON THE Y AXIS. IF NO COLUMN IS SELECTED (THE DEFAULT), ALL COLUMNS WILL BE PLOTTED ON THE Y AXIS AGAINST THE CONSECUTIVE INTEGERS ON THE X AXIS.

PARAMETER 10 IS THE NAME OF THE VARIABLE CONTAINING THE MATRIX OR VECTOR TO BE PLOTTED. VECTORS ARE PLOTTED AGAINST THE CONSECUTIVE INTEGERS. COLUMNS CORRESPOND TO CURVES.

PARAMETER 11 IS THE NAME OF THE VARIABLE TO RECEIVE THE GRAPH. THE DEFAULT, NO VARIABLE, DISPLAYS THE GRAPH ON THE SCREEN.

An example illustrating the use of EZPLOT follows. We begin by creating a data vector which will be plotted.

```
X 1.LO.LO(%15)#.IO60 " SINE CURVE
EZPLOT
```

EZPLOT

1. REVIEW/ENTER/CHANGE SETTINGS
2. GENERATE GRAPH
3. HELP
4. EXIT

ENTER NUMBER

.BX:

1

	DEFAULT	CURRENT
--	---------	---------

- | | | |
|--|------|------|
| 1. NUMBER OF VERTICAL DIVISIONS | 21 | 21 |
| 2. NUMBER OF HORIZONTAL DIVISIONS | 60 | 60 |
| 3. TITLE | | |
| 4. X AXIS LABEL | | |
| 5. Y AXIS LABEL | | |
| 6. LOGARITHM ON X AXIS | NO | NO |
| 7. LOGARITHM ON Y AXIS | NO | NO |
| 8. PRINT CHARACTERS | 0*#+ | 0*#+ |
| 9. COLUMN NUMBER CORRESPONDING TO X AXIS | | |
| 10. INPUT DATA VARIABLE | | |
| 11. OUTPUT (GRAPH) VARIABLE | | |

ENTER NUMBER OF ITEM TO SET/CHANGE OR 0 (ZERO)
TO RETURN TO MENU

.BX:

10

ENTER M TO RETURN TO MENU WITHOUT SETTING/CHANGING ITEM.
ENTER D TO RESTORE DEFAULT.
ENTER VALUE OR TEXT (WITHOUT QUOTES) TO SET/CHANGE ITEM.

X

ENTER NUMBER OF ITEM TO SET/CHANGE OR 0 (ZERO)
TO RETURN TO MENU

.BX:

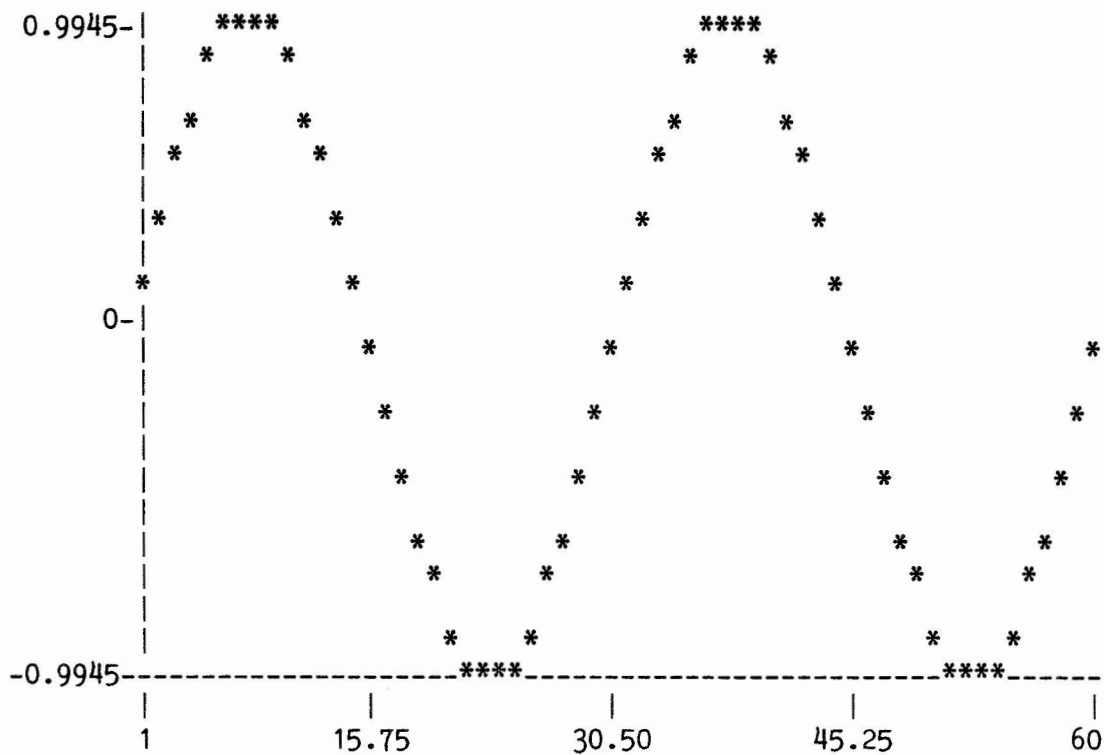
0

1. REVIEW/ENTER/CHANGE SETTINGS
2. GENERATE GRAPH
3. HELP
4. EXIT

ENTER NUMBER

.BX:

2



- 1. REVIEW/ENTER/CHANGE SETTINGS
- 2. GENERATE GRAPH
- 3. HELP
- 4. EXIT

ENTER NUMBER

.BX:
4


```

VEZPLOT;C;NO;VAL;VDIV;HDIV;COL;TITLE;XLABEL;YLABEL;VLOG;HLOG;CHARS;INPUT;
OUTPUT
[1]  RWRITTEN BY PAUL KOBRIN, OCTOBER 4, 1982.

[2]  '                               EZPLOT'
[3]  MENU:
[4]  ' '
[5]  '1. REVIEW/ENTER/CHANGE SETTINGS'
[6]  '2. GENERATE GRAPH'
[7]  '3. HELP'
[8]  '4. EXIT'
[9]  ' '
[10] 'ENTER NUMBER'
[11] →(1=NO←□)/REVIEW
[12] →(2=NO)/GEN
[13] →(3=NO)/HELP
[14] →(4=NO)/O
[15] 'TRY AGAIN'
[16] →MENU
[17] REVIEW: ' '
[18] ' '                               DEFAULT                               CURRENT'
[19] ' '
[20] C←'21'
[21] →(2≠□NC 1 4p'VDIV')/L1
[22] C←VDIV
[23] L1:'1. NUMBER OF VERTICAL DIVISIONS'           21                               ',C'
[24] C←'60'
[25] →(2≠□NC 1 4p'HDIV')/L2
[26] C←HDIV
[27] L2:'2. NUMBER OF HORIZONTAL DIVISIONS'       60                               ',C'
[28] C←' '
[29] →(2≠□NC 1 5p'TITLE')/L3
[30] C←TITLE
[31] L3:'3. TITLE'                                ' '                               ',C'
[32] C←' '
[33] →(2≠□NC 1 6p'XLABEL')/L4
[34] C←XLABEL
[35] L4:'4. X AXIS LABEL'                         ' '                               ',C'
[36] C←' '
[37] →(2≠□NC 1 6p'YLABEL')/L5
[38] C←YLABEL
[39] L5:'5. Y AXIS LABEL'                         ' '                               ',C'
[40] C←'NO'
[41] →(2≠□NC 1 4p'HLOG')/L6
[42] C←'YES'
[43] L6:'6. LOGARITHM ON X AXIS'                   NO                               ',C'
[44] C←'NO'
[45] →(2≠□NC 1 4p'VLOG')/L7
[46] C←'YES'
[47] L7:'7. LOGARITHM ON Y AXIS'                   NO                               ',C'
[48] C←'O*x+o∇Δ>cU'
[49] →(2≠□NC 1 5p'CHARS')/L8
[50] C←CHARS
[51] L8:'8. PRINT CHARACTERS'                       O*x+o∇Δ>cU                       ',C'
[52] C←' '
[53] →(2≠□NC 1 3p'COL')/J9
[54] C←COL
[55] J9:'9. COLUMN NUMBER CORRESPONDING TO X AXIS' ' '                               ',C'
[56] C←' '
[57] →(2≠□NC 1 5p'INPUT')/L9
[58] C←INPUT
[59] L9:'10. INPUT DATA VARIABLE'                  ' '                               ',C'
[60] C←' '
[61] →(2≠□NC 1 6p'OUTPUT')/L10
[62] C←OUTPUT
[63] L10:'11. OUTPUT (GRAPH) VARIABLE'              ' '                               ',C'
[64] SET:
[65] ' '

```

```

[66] 'ENTER NUMBER OF ITEM TO SET/CHANGE OR 0 (ZERO)'
[67] ' TO RETURN TO MENU'
[68] NO←□
[69] →((NO>LNO)∨(NO<[NO])∨(NO<1)∨(NO>11))/MENU
[70] ' '
[71] 'ENTER M TO RETURN TO MENU WITHOUT SETTING/CHANGING ITEM.'
[72] 'ENTER D TO RESTORE DEFAULT.'
[73] 'ENTER VALUE OR TEXT (WITHOUT QUOTES) TO SET/CHANGE ITEM.'
[74] ' '
[75] VAL←,□
[76] →(('M'=1+VAL)∧1=ρVAL)/MENU
[77] →(I1,I2,I3,I4,I5,I6,I7,I8,K9,I9,I10)[NO]
[78] →MENU
[79] I1:→(VAL='D')/I11
[80] VDIV←VAL
[81] →SET
[82] I11:C+□EX 'VDIV'
[83] →SET
[84] I2:→(VAL='D')/I21
[85] HDIV←VAL
[86] →SET
[87] I21:C+□EX 'HDIV'
[88] →SET
[89] I3:→(('D'=1+VAL)∧1=ρVAL)/I31
[90] TITLE←VAL
[91] →SET
[92] I31:C+□EX 'TITLE'
[93] →SET
[94] I4:→(('D'=1+VAL)∧1=ρVAL)/I41
[95] XLABEL←VAL
[96] →SET
[97] I41:C+□EX 'XLABEL'
[98] →SET
[99] I5:→(('D'=1+VAL)∧1=ρVAL)/I51
[100] YLABEL←VAL
[101] →SET
[102] I51:C+□EX 'YLABEL'
[103] →SET
[104] I6:→(('D'=1+VAL)∧1=ρVAL)/I61
[105] HLOG←'YES'
[106] →SET
[107] I61:C+□EX 'HLOG'
[108] →SET
[109] I7:→(('D'=1+VAL)∧1=ρVAL)/I71
[110] VLOG←'YES'
[111] →SET
[112] I71:C+□EX 'VLOG'
[113] →SET
[114] I8:→(('D'=1+VAL)∧1=ρVAL)/I81
[115] CHARS←VAL
[116] →SET
[117] I81:C+□EX 'CHARS'
[118] →SET
[119] K9:→(('D'=1+VAL)∧1=ρVAL)/K91
[120] COL←VAL
[121] →SET
[122] K91:C+□EX 'COL'
[123] →SET
[124] I9:→(('D'=1+VAL)∧1=ρVAL)/I91
[125] INPUT←VAL
[126] →SET
[127] I91:C+□EX 'INPUT'
[128] →SET
[129] I10:→(('D'=1+VAL)∧1=ρVAL)/I101
[130] OUTPUT←VAL
[131] →SET
[132] I101:C+□EX 'OUTPUT'
[133] →SET
[134] GEN:
[135] →(2=□NC 1 ρ'OUTPUT')/N1
[136] PLOT ←INPUT
[137] →MENU

[138] N1:
[139] C+2 27ρ'FUNCTION
[140] C[2;:ρOUTPUT]←OUTPUT
[141] ←□FX C
[142] □SINK←□EX 1 8ρ'FUNCTION'
[143] →MENU
[144] HELP:
[145] EZPLOTHOW
[146] →MENU
▽
*****
VZ←CENTER X
[1] Z←((L4.5-0.5×ρZ)ρ' '),Z,(L4.5-0.5×ρZ+FORMAT X)ρ' '
▽
*****
VZ←FORMAT X;U
[1] Z←0
[2] →((1000>|X)∧X=[X])/L1
[3] Z←0+((3-L10×|X)×(1000>|X)∧0.001≤|X)-3×(0.001>|X)∨10000000≤|X
[4] L1:Z←(9,Z)∨(X×1000≥|X)+(1000<|X)×U×e6 0∨X±U+10*-3+L10×(X=0)+|X
[5] Z←(Zρ' ')/Z
▽
*****
VZ←HCOMBINE X;N;I
[1] N←ρX←,X
[2] Z←' '
[3] I←1
[4] L1:Z←Z,CENTER X[I]
[5] →(N≥I+I+1)/L1
▽
*****
VZ←RTJUST X
[1] Z←((9-ρZ)ρ' '),Z+FORMAT X
▽
*****
VZ←VCOMBINE X;N;I
[1] N←ρX←,X
[2] Z←' '
[3] I←1
[4] L1:Z←Z,RTJUST X[I]
[5] →(N≥I+I+1)/L1
▽

```

LARGE DATA BASE HANDLING WITH APL-NET,
A HIGH-PERFORMANCE APL SYSTEM
FOR THE DECSYSTEM-20

by Jean-Pierre Barasz
Business Applied Real Time Systems (BARTS)
54, rue d'Amsterdam
75009-Paris, FRANCE

Abstract

APL systems are rarely used as a support for large data bases, for efficiency reasons. This paper will attempt to show how a powerful file system and a well designed set of APL extensions can provide a means to efficiently create, maintain and query large data bases.

Introduction

Handling large data bases in APL (up to several hundred million bytes) involves 2 different processing phases:

- the loading phase, which typically involves reading and processing sequential files generally produced in non-APL environments, inverting the file, selecting data, numeric conversion etc..., and storing the reorganized data in some APL-oriented file structure designed to facilitate the query and/or the maintenance of the data base.
- accessing the data base, possibly with simultaneous updating and querying.

This paper presents selected features of APL-NET, a high-performance APL system for the DECSYSTEM-20, and shows how they can be used effectively in these 2 phases of large data base handling.

Interpreter Performance

APL has an undeserved notoriety for being slow compared to compiled languages, as it is interpreted. Although an APL system must analyse statements before executing the machine instructions that perform the prescribed functions - whereas a compiled system separates the 2 phases and does the analysis and code production phase only once - the parallel array processing characteristic of APL can in many cases completely eliminate this disadvantage. In addition, there are important variations in the internal speed of different APL system implementations, as well as in the performance of different language compilers. Speed ratios of 2 or 4 to 1 are not uncommon.

Parallel Processing

The key to efficient processing in APL is processing large chunks of data in parallel. In order to execute a primitive function, for example $A + B$, an APL system will do the same amount of work to analyse the statement (and possibly, in some implementations, produce machine code), whatever the size of A and B. The actual machine resources needed to execute the addition function, however, are proportional to the size of the result of the operation. The "work" involved in analysing the expression, checking for argument conformability, obtaining space for the result, etc. in a good APL interpreter represents 200 to 300 machine instructions. In some implementations, it may be a good deal more. The work in doing the actual addition of two elements is typically 4 to 10 machine instructions. An easy computation shows that in adding 2 scalars, the interpretation cost may be 100 times the actual execution cost, in the worst case, whereas in adding two 1000-element arrays, the interpretation cost may be only a negligible percentage of actual execution cost.

Although most APL systems have relatively good performance in the interpretation of some primitive functions, less care has been given to the power and efficiency of such features as file systems. In applications where files are small or file access rare, the efficiency of file systems is not important, but the weaknesses of such systems prohibits their use when the amount of data gets large.

Background on APL File Systems

Relatively early in the evolution of APL systems, the need for a means to store data external to the workspace arose and was met by file systems such as APL*PLUS (trade mark of S.I.S.C.), and I.P. SHARP's. These file systems, which have served as models for many of the file systems developed since, gave only partial solutions.

In a manner conforming with APL "philosophy", those file systems did away with "job control language", logical or physical disc space allocation and various other esoteric system-dependent features. They allowed the storage of APL arrays with all their attributes (rank, type and dimensions), concurrent update access by several users, and sophisticated access protection mechanisms.

Despite their qualities, these file systems had two important weaknesses: the addressing of data components of APL files, and the means to access "non-APL" files produced by other systems.

APL-NET File System and Extensions

APL-NET allows handling of large data bases, thanks to its key system for APL files, various other enhancements of the file system and other system functions that allow efficient parallel processing of large chunks of data.

Keyed Files vs. Sequential Component Files

In "sequential component" file systems, the addressing of the data stored in the file is through component numbers, a set of contiguous integers starting at 1. It is not possible to delete file components other than at the beginning or end of the file; applications must replace deleted components with empty arrays, although this technique only partially releases file space.

The component number can rarely be used directly as a meaningful "key" into the data base. This means that access mechanisms such as indexes, hash-coding, etc. must be written in APL code. Maintaining file indexes or keys at the application level, in APL code, has two major drawbacks:

- inefficiency since the cost of accessing one component grows rapidly as a function of the total number of components, since APL code must be interpreted for each file access.
- fragility in case of system crashes, where the application-maintained indexes may not correctly reflect the state of the data in the file in case it was being updated at the time of crash.

APL-NET provides a much more sophisticated file system for APL components, called "keyed APL component files". As in the APL*PLUS file system, the file is made up of "components" containing APL arrays. The components are time-stamped and signed with the user number of the last component writer. File accesses are controlled with an access control matrix specifying individual access rights for each authorized user or user group. The file system is internally interlocked to allow simultaneous multiple user update. It also provides an interlocking mechanism at the user level as a system function. Where the APL-NET file system differs from APL*PLUS is in the "key" system and the support functions. To each component is associated a key. The key is either a 1- to 13-digit integer scalar, an integer vector of less than 30 elements, or a character vector of less than 120 elements.

In addition to the system functions that allow reading, writing or removing (multiple) components, APL-NET provides functions to manipulate and count the keys, give subsets of keys between specified boundaries, query the neighbors of a given key, sequential access, appending, etc.. All in all, 30 system functions are associated with file management.

The "keyed" file system allows users to directly implement complex access schemes by using the component key as meaningful data (part number, customer number, etc...). The (internal) management of the keys and data has no inherent limitations other than the physical disc space. There is no "reservation" of space for the keys or data. Disc space is released to the system as components are removed or replaced by smaller ones.

Applications using the keyed file system are simpler to write and maintain, more secure, and more cpu efficient than comparable ones using the less sophisticated APL*PLUS-like file systems.

External Files (Non-APL)

We have seen how the APL-NET keyed files can serve as a support for large data bases, but before using a data base, it must be built from data files generally "external" to the APL system.

To build the APL files from external files, one must have the possibility either to access external files from APL, or to access APL files from other environments (COBOL, FORTRAN, etc.). In most APL systems that allow access to external files from APL, the process is generally too expensive for large data bases, as reading and processing records one at a time turns out to be exactly the kind of process where APL is not very efficient. In systems where APL files may be accessed and built from other languages, it is possible to write the data base loading programs in another language, although such programs are often "use-once and modify" type of programs, typically what APL is supposed to be good at, but not "batch-oriented languages". The machine efficiency is traded for programmer inefficiency.

APL-NET has solved the external file access problem by providing the following features.

At the choice of the programmer, an external file may be viewed as a sequential collection of fixed-length bit, word, ASCII-character records, as a set of (possibly non-contiguous) pages, or as variable length ASCII-character records terminated by line-feed characters. The way a file is viewed and record length (where applicable) is determined is through the parameters of the single system function `□FOPEN`, which is used to create, open, or supersede an existing file. A file may be simultaneously open in different modes on several channels.

Accessing data may be sequential or direct, even for variable length ASCII files! An arbitrary number of records may be read (or written) in a single call to the `□FREAD` (or `□FWRITE`) system function(s). Reading multiple records yields a matrix result, except for variable length ASCII files, where a vector containing embedded "new-line" characters is returned.

End-of-file processing is easily performed, since the `□FREAD` function will return a "short" matrix or vector if fewer than the requested number of records exist beyond the requested first record. The powerful APL-NET error trapping mechanism may be effectively used to detect and process the end-of-file condition, which sets an error event if one attempts to read sequentially after having reached the end of the file.

Hence,

```
1000 □FREAD 1 0
```

will read a thousand (or fewer) records from the file opened on channel 1 at the current file position, and

```
10 □FREAD 1,I
```

will read 10 records starting with record number I.

Example of control logic for processing a sequential 80-character record ASCII file "chunks" at a time:

```

VLOAD FILE;BLOC;DATA
[1] 'CARDS.FILE.3' □FOPEN 1 0 ~7 80
[2]           A OPEN FILE 'CARDS.FILE.3'
[3]           A NOTE FULL TOPS-20 NAMING
[4]           A 1: CHANNEL 1
[5]           A 0: EXISTING FILE
[5]           A ~7: ASCII CHARACTERS
[6]           A 80: RECORD LENGTH IS 80
[7] 58 □TRAP EOF A ERROR TRAP TO LABEL 'EOF'
[8]           A ON END-OF-FILE CONDITION
[9] BLOC*1000   A ARBITRARILY LARGE BLOC
[10]          A SIZE FOR EFFICIENT
[11]          A PARALLEL PROCESSING
[12] LOOP: DATA*BLOC □FREAD 1 0
[13]          A READ BLOC AT CURRENT
[14]          A FILE POSITION: YIELDS
[15]          A 1000 BY 80 MATRIX
[16] PROCESS DATA ◇ →LOOP
[17] EOF: □FCLOSE 1 A CLOSE FILE
V

```

Reading records "chunks" at a time is 10 time faster in APL-NET than the fastest APL program that reads records one at a time to build the matrix!

Processing Variable Length Record ASCII Files

APL-NET has numerous system functions which may be useful to efficiently process variable length ASCII files. In many cases, the input file is composed of heterogeneous records, perhaps of differing length. Without the system functions discussed below, it would be difficult to process such a file in large chunks to benefit from APL's inherent efficiency whenever parallel processing can be used.

As discussed before, APL-NET permits reading several records at one time from a variable length record ASCII file with a single call to the □FREAD system function. This yields a character vector with embedded "new-line" characters as record delimiters. APL-NET provides a system function, □STACK, and its inverse, □UNSTACK, which can be used to transform a vector into a matrix and vice-versa, by specifying delimiting and filling characters. The delimiting character is the character which separates individual "records", and the filling character is used to pad on the right the matrix rows made up of "short" records. Example:

```

A←'/*' □STACK 'JOHN/SAM/PETER/ELSA'
A
JOHN*
SAM**
PETER
ELSA*
ρA

```

4 5

Using the "new-line" and "space" characters as delimiting and filling characters, respectively, allows breaking up the records of a variable length ASCII file into a matrix which can be processed efficiently. Example:

```

ρDATA←'␣ ' □STACK 1000 □FREAD 1 0
1000 125
if the longest record is 125 characters long.

```

□STACK is 7 times faster than a similar routine written in APL.

A frequent problem with loading a data base is the selection of specific records for appropriate processing. APL-NET offers a system function, □SEARCH, used for string searches, which accepts a data argument of arbitrary rank. The search argument, a vector, is looked up in each row of the data argument. The result is of the same shape as the data argument and contains a 1 one in the corresponding leftmost position(s) of the data argument where the search string is matched. Appropriate processing of the result of the search yields a powerful selection mechanism. Example:

```

A
JOHN*
SAM**
PETER
ELSA*
A □SEARCH 'SA'
0 0 0 0 0
1 0 0 0 0
0 0 0 0 0
0 0 1 0 0

```

□SEARCH may in many cases bring a significant increase in efficiency in data base loading and other selection processes.

Another frequent problem with data base loading is the numerizing of fields from the source file. APL-NET provides a system function for numeric conversion, □NUMERIC, which works with a field description and a character matrix data argument, yielding the numeric values of the fields as result. Since the data is known to be homogeneous within the fields, the conversion algorithms are particularly efficient.

Conclusion

A fast APL interpreter, with a good set of extensions including a powerful file system, is capable of running applications handling large amounts of data, provided care is taken to process data in parallel whenever possible. Such applications may run as well, if not better, than applications written in less powerful and user-friendly languages.

ADVERTISING UNLIKELY

Douglas Bohrer
 Editor, The Special Character Set

As the culmination of the "renegade editors'" advertising investigation, I drafted a policy and questionnaire for distribution in Cincinnati. I showed advance copies to Mike York, DECUS Newsletter Coordinator, and the members of the ad hoc advertising committee. After making the changes they suggested, I had several hundred printed at my own expense. During the leadership conference on the Sunday before sessions began I showed the survey to several people including Charles Mustain, incoming Executive Board member and former Publications head. There seemed to be agreement that it was OK to pass it out.

Susan Abercrombie, an APL SIG Steering Committee member, was in charge of the DECUS Library booth on the display floor. She began distributing surveys at the booth on Monday morning.

The DECUS Management Council banned the questionnaire on Monday afternoon. They did not have a copy of the questionnaire at the meeting, nor did they ask for any input from Sue, me or anybody else. The stated reason for totally prohibiting further distribution was that the questionnaire looked "officially sanctioned" and some members would mistakenly assume that the adoption of the proposed policy was all but inevitable.

I tried passing out the questionnaire on Wednesday as I walked through the halls and had an outgoing DECUS Board member threaten me with expulsion from the symposium if I continued. As a result of all this activity, there were only 64 responses, which are summarized below.

The combination of negative board reaction and sizeable negative feedback on the responses to the questionnaire make any change in the current advertising policy extremely unlikely.

Many people I talked to thought that the action banning the survey was ill considered. I have reprinted the questionnaire so that you can judge for yourself. While the Management Council and Executive Board have the power to censor anything they choose to, I feel that banning independent questionnaires on DECUS policy issues will stifle most organizational change. Procedures specifying what steps are needed to approve distributing such questionnaires would be useful for the future.

ADVERTISING QUESTIONNAIRE RESULTS

STRONGLY AGREE				STRONGLY DISAGREE				NO RE	Q1 TOTAL	
Q2 A1	Q2 A2	Q2 A3	Q2 A4	Q2 A5	Q2 A6	Q2 A7				
9	0	1	0	1	3	2	0	16	Q1 A1	STRONGLY AGREE
3	2	2	0	0	1	0	0	8	Q1 A2	
3	2	1	0	0	0	0	0	6	Q1 A3	
1	2	0	2	0	0	0	0	5	Q1 A4	
1	0	0	0	0	0	1	0	2	Q1 A5	
4	1	0	1	0	2	0	0	8	Q1 A6	
5	0	1	1	1	0	8	3	19	Q1 A7	STRONGLY DISAGREE
26	7	5	4	2	6	11	3	64	COLUMN TOTALS	
AS A PERCENTAGE OF THE 64 RESPONSES										
14.06	0	1.56	0	1.56	4.69	3.12	0	25	Q1 A1	STRONGLY AGREE
4.69	3.12	3.12	0	0	1.56	0	0	12.5	Q1 A2	
4.69	3.12	1.56	0	0	0	0	0	9.37	Q1 A3	
1.56	3.12	0	3.12	0	0	0	0	7.81	Q1 A4	
1.56	0	0	0	0	0	1.56	0	3.12	Q1 A5	
6.25	1.56	0	1.56	0	3.12	0	0	12.5	Q1 A6	
7.81	0	1.56	1.56	1.56	0	12.5	4.69	29.69	Q1 A7	STRONGLY DISAGREE
40.62	10.94	7.81	6.25	3.12	9.37	17.19	4.69	100	COLUMN TOTALS	

DRAFT NEWSLETTER ADVERTISING POLICY

The following policy has been proposed to allow commercial paid advertising in DECUS Newsletters. The revenue would be used to lower subscription fees. Your input will be helpful in considering whether or not this or any policy change with regard to advertising should be adopted. (So far only a few renegade editors have endorsed this proposal.)

PROPOSED POLICY

1. A special exemption to the non-commerciality guidelines is granted to newsletter advertising sections. Advertising sections must be entirely separate from editorial content. The current policy banning commercialism remains in effect for all editorial content.
2. Acceptable advertising will contain information about products that work on or with Digital Equipment Corporation products. The products advertised may be Digital made or competing products as long as they are usable with some Digital made product. Newsletter editors are responsible for accepting advertising appropriate to their publications. Advertisers may appeal any exclusion to the Management Council.
3. Advertising fees will be set annually by the SIG(s) sponsoring each newsletter with the approval of the Management Council. The revenue will be used to offset production costs and lower subscription prices for newsletters. Money for advertisements must be paid in advance of the publication deadline for the newsletter direct to DECUS.
4. Foreign chapters, who reprint newsletters, will have the option of negotiating the inclusion of advertisements in their reprints or removing the advertising section.

QUESTIONNAIRE

1. DECUS newsletters should accept advertising.

Strongly Agree							Strongly Disagree
1	2	3	4	5	6	7	

2. A pilot program with one or two small newsletters taking advertising should be tried before adopting an advertising policy for all newsletters.

Strongly Agree						Strongly Disagree
1	2	3	4	5	6	7

3. What changes in the proposed policy would make it more acceptable? What comments would you like to make on accepting advertising?



DECUS
EUROPE

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY

P.O. Box 510 CH-1213 Petit-Lancy 1, Geneva Switzerland Tel. (022) 93 33 11 Telex 422 593

Geneva
27 July 84

To: Douglas Bohrer
Bohrer & Company
903 Ridge Road, Suite 3
Wilmette
Illinois 60091

Dear Douglas,

Here are the European newsletter distribution figures I promised to send you. When reading them, remember that, for the moment, they are free to the membership (paid by DEC still) in Europe. We are currently testing our subscription system and members will have to start paying for them in January next year, we expect.

Regarding your questions about accepting advertising inside SIG newsletters, I remain sceptical about the practical aspects of it, as I mentioned to you in Cincinnati. Most of my European colleagues to whom I have spoken are in broad agreement with me, although a few like the idea in principle. However, as I said in June, as long as any adverts are included as an "appendix" in the newsletters as you had proposed, we in Europe would be free to make our own choice at that time. The only thing I would ask of you is that we are kept informed of any change with sufficient warning that the European board can discuss the matter in good time.

Yours sincerely

Alan Silverman
(European SIG Coordinator)

Copies to

Mark Grundler (DECUS US)
Mike York (DECUS US)
Ralf Broeck (DECUS Europe Chairman)
Otto Titze (DECUS Europe deputy SIG Coord.)
Gaby Neukom (DECUS Europe SIG Secy)
Dale Brandt (DECUS Europe Administrator)

NUMBER OF EUROPEAN SUBSCRIBERS BY US SIG NEWSLETTER

<u>Newsletter</u>	<u>Total Subscribers</u>
LABS/SITE/HMS	2,168
GRAPHICS/COBOL/DIBOL	2,098
APL	157
UNISIG	1,112
RSTS-Cache Buffer	1,026
EDUSIG	695
MUMPS SIG	731
STRUCTURED LANG. (to be Languages and Tools) "The Heap"	2,905
Networks - "Networks"	2,078
DATATRIEVE "Wombat Ex"	993
AT LARGE "At Large"	478
BASIC SIG	1,514
VAX "Pageswapper"	4,059
RT-11 SIG "Minitasker"	2,696
RSX-IAS SIG "Multitasker"	4,404
<u>TOTAL</u>	<u>27,114</u>

NOTE: These are the CURRENT numbers. When we go to paying subscribers, we anticipate a 25% return.

Date: Fri 31-AUG-1984 11:44
From: Judy Arsenault

TO: Doug Bohrer (BOHRER)

Subject: SUB. SERVICE INFORMATION

REPORT ON SUBSCRIPTION SERVICE FOR FY'85-AUGUST

The following is a report on each offering showing subscriptions taken in FY'84 that have not yet expired and New/Renewal subscriptions for FY'85. I will provide a legend to explain each column of figures.

Column 1: FY'84 subscribers still active on system
Column 2: FY'85 new/renewed subscribers
Column 3: Total # of Subscribers per offering
Column 4: Total Number of subscribers at this time last year

Please keep in mind that some of the offerings have split from the FY'84 original offerings.

OFFERING	COLUMN 1 FY'84	COLUMN 2 FY'85	COLUMN 3 T.Subscribers	COLUMN 4 Comparison
APL	232	447	679	652
BAS	635	1269	1904	1816
DIC*		839	1584	
OAG*		1417	2162	
*(OLD OAD)	745			1778
DTR	771	1389	2160	1912
EDU	397	720	1117	1028
IAS*		405	1354	
RSX*		1911	2860	
*(OLD RSX)	949			2922
LGS	327	596	923	865
LHS	764	1171	1935	1699
LTS*		1371	1905	
MMP*		495	1029	
*(OLD MSL)	534			1288
NTW	934	1734	2668	2317
RST	475	943	1418	1458
RT	699	1537	2236	2322
UNI*		852	1296	
*(OLD SOS)	444			1132
VAX	1739	3286	5025	4089

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY



DECUS
AUSTRALIA

7 May 1984

Dear Doug,

Your letter regarding newsletter advertising arrived on the very day I left for a Chapter Board meeting. You could not have timed it better!

First I must express, both my personal thanks for your inquiry, and also that of the Decus Australia Board. I have worked for many years to establish real communication between Decus members worldwide and at last we do appear to be getting somewhere. I believe that my visit to Las Vegas was the most productive that I have ever made. Whilst I was there, I was made to feel welcome by the US chapter, and the events which have followed have shown the sincerity of all involved.

The Board gave careful consideration to your proposals and give your ideas full support. The idea of incorporating adverts into a special section at the rear of the publication is excellent in that it leaves other chapters free to publish or not depending of the relevance of the advert. We decided that we would be willing, at least in principle, to reprint any adverts, but that we would do so without requiring any part of the advertising fee.

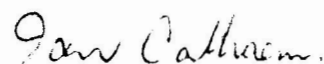
Our numbers are of course much lower than yours and are still building up again after the introduction of a subscription fee. Current numbers are:-

APL	13	SS & OS	35
BASIC	143	STR LANG.	109
DATATRIEVE	121	LABS/SITE MGMT	97
EDUSIG	53	RT11	148
LARGE SYSTEMS	26	QA/GRAPHICS	135
NETWORKS	132	RSX/IAS	140
RSTS	104	VAX/VMS	232

One question which you might be able to follow up for us concerns the cost of the bromides which we receive to make our reports from. We have just received the first account from the US Chapter and based on local prices, the cost is unbelievably high. (some 400-500% higher than would dreamt of!) I work in this field at present setting up phototypesetting systems for the government so I believe I have a realistic approach to costing, and I cannot even start to guess why the charges are so high. I hope that the US Chapter made a typing error when sending the account.

Again our sincere thanks for thinking about us and I hope we can continue to all work for the benefit of Decus worldwide.

Regards,



SURVEY RESULTS

Here are the results of the survey published in the last issue. There were a total of 9 responses. If it hadn't been for all the articles you guys sent in the underwhelming response rate would have destroyed the editor's morale. We got 25 responses from the questionnaire in issue number 1, when our circulation was about 10 percent of what it is now.

1. I use the following computers (check all that apply):

<input type="checkbox"/> 2_PDP-11/03 LSI-11 (Q-BUS)	<input type="checkbox"/> MINC
<input type="checkbox"/> 2_PDP-11/23 11/23+ (Q-BUS)	<input type="checkbox"/> Q-BUS 68000
<input type="checkbox"/> 3_Rainbow 100 series	<input type="checkbox"/> 1_Professional 300 series
<input type="checkbox"/> 5_VAX-11	<input type="checkbox"/> Micro-VAX
<input type="checkbox"/> 1_DEC-20	<input type="checkbox"/> DEC-10
<input type="checkbox"/> NONE OF THE ABOVE	
<input type="checkbox"/> OTHER 11/44 11/34A 11/24 (2)11/70 11/45 PDP-8	

2. I have access to the following media (check all that apply):

<input type="checkbox"/> 6_5.25 inch floppy	<input type="checkbox"/> RL02
<input type="checkbox"/> 6_8 inch RX01 floppy single density	<input type="checkbox"/> 2_8 inch RX02 floppy double density
<input type="checkbox"/> 6_0.5 inch magnetic tape 800 BPI	<input type="checkbox"/> 7_0.5 inch magnetic tape 1600 BPI
<input type="checkbox"/> 3_0.5 inch magnetic tape 6250 BPI	
<input type="checkbox"/> Other portable media_(2)RL01 RK07 RX50K	

3. I use the following operating system(s) (check all that apply and fill in version):

<input type="checkbox"/> 3_RT-11 version 3B,4,5	<input type="checkbox"/> 1 TSX+ version 5
<input type="checkbox"/> RSX-11 version_____	<input type="checkbox"/> 1 RSX-11M version_4.1
<input type="checkbox"/> 3 RSTS/E version 8.0	<input type="checkbox"/> UNIX version_____
<input type="checkbox"/> 5 VAX/VMS version 3.5	<input type="checkbox"/> TOPS 10 version_____
<input type="checkbox"/> 1 TOPS 20 version 5.1(5050)	
<input type="checkbox"/> Others CPM-86 IASv3.1 P/OSv1.7 OS8	

4. I use the following APL's (check all that apply):

<input type="checkbox"/> 2 APL-11 V1 (RT-11)	<input type="checkbox"/> 1 APL-11 V 2 (RT-11)
<input type="checkbox"/> APL-11 V2 (RSX-11)	<input type="checkbox"/> 2 APL-11 V 1 (RSTS/E)
<input type="checkbox"/> 4 VAX-11 APL	<input type="checkbox"/> 1 APL-SF (TOPS-10,TOPS-20)
<input type="checkbox"/> 1 None of the above	
<input type="checkbox"/> Others BURROUGHS APL V3.1	

5. I do most of my APL work using (pick one):

<input type="checkbox"/> 4 APL special character set	<input type="checkbox"/> 5 TTY mnemonics
--	--

6. This newsletter should accept paid advertising (circle one):

strongly agree

strongly disagree

1
5

2
1

3
1

4
1

5

6
1

7

7. What do you want the APL SIG to do that it isn't?

(2)MORE PARTICIPATION FROM MEMBERS

(2)MORE VAX-11 PARTICIPATION

MORE SYSTEM INDEPENDENT HINTS

(3)MORE PARTICIPATION FROM DEC, ESPECIALLY NEWS ARTICLES

MAKE A BETTER APL-11

GET AN APL FOR THE RAINBOW

MICRO-VAX APL

FULL SCREEN APL EDITOR FOR THE VAX

MORE LIBRARY SUBMISSIONS

8. What is the APL SIG doing that you think we should stop?

NO RESPONSES

9. What should the APL SIG continue doing ?

(2)NEWSLETTER

SYMPOSIA PRESENTATIONS

EXISTING

THE SPECIAL CHARACTER SET PAGE 22 OCTOBER 15, 1984

15 October 1984

Dear APL Fanatic:

I thank all of you who sent me material for this issue. It was great to have so many varied contributions from all over the world. The next issue is totally empty. This worries me. Please, PLEASE, write me some more articles so that we can keep this newsletter alive.

Almost anyone can write an article. It doesn't have to be long or heavy. Write an article about how your site uses APL or/and how it chose APL in the first place. Write an article about that handy function you just wrote. Write a testimonial about how APL changed your life.

Maybe you already have an article on an APL topic, but you published it somewhere else. Let me republish it if you can give me copyright permission.

I have printed a style sheet for any article you write. However, even if your article is in another format I'll take it. I can read RT-11 RX02 floppies. I have a volunteer typist. JUST GIVE ME SOMETHING TO PRINT!

The deadline for the next issue is 15 February 1984. However, if you have something now SEND IT NOW. The sooner I get your article, the better I'll sleep nights after worrying about the next issue.

Sincerely,



Doug Bohrer
Bohrer and Company
903 Ridge Road, Suite 3
Wilmette, IL 60091 USA
Phone: 312-251-9449

