systems would be most greatly facilitated, and APL source programs could be processed (key-punched, edited, updated, copied, etc.) by existing non-APL facilities. This is clearly not a very revolutionary proposal, as practically every other language processor I can bring to mind offers some similar facility.

APL systems could also provide facilties for more efficient non-symbolic dumping and restoring of user programs (and data, perhaps) at a single installation. I think there are some important drawbacks, however, using a dump-restore facility as a medium for program exchange. The physical format of this sort of dump is usually chosen in an attempt to optimize efficiency under conditions existing in a particular system. It is probably inevitable that these conditions will vary and diversity rapidy when more than one system is involved. Adopting a standard non-symbolic dump format for a wide range of unlike systems would result in less optimum efficiency and occasional incompatibility, and so I would expect any such standard to be generally ignored. Furthermore, the potential for confusion inherent in program exchange is considerable, and so choosing an exchange medium to help reduce that confusion is most desireable. Any format which involves its own unique syntax rules can only serve to compound the confusion.

Card-image format is a simple and general medium for practically any program exchange, and it offers the best hope for easy analysis of difficulties when they arise. In any case, I don't think we can realistically expect any standard other than symbolic 80-column card format to be widely adopted.

Modifications to the APL 1130/System to Provide More Convenient Operating on a Fortran User's Machine

Part I

J. F. Clementi
IBM (Australia),
95 North Terrace,
Adelaide. 5000
South Australia

R. P. Fletcher,
School of Mathematical Sciences,
The Flinders University of South Australia,
Bedford Park. 5042
South Australia

This year APL/1130 is being introduced at Flinders University for use as a second language to FORTRAN. It is being taught to a small group of second year undergraduates, who will use it, rather than FORTRAN, for their programming in an introductory Numerical Analysis course, and to other persons wishing to use APL/1130 as a second (or first) language.

Since FORTRAN is the University's main language and since the machine (an 8k, 3.6 microsecond, machine, with single disk drive, 1132 printer and 1442 card reader/punch) carries a heavy load, it is imperative that day-to-day APL computing be done via cards. This led us to look at possibilities of:

i Improving throughput
ii Making the system more convenient for the operators
iii Making the system more convenient for the users

To date we have achieved the following successes:

(a) CARD OPTION DEFAULTS

The standard APL/1130 system provides that in card mode, unless otherwise specified, the options EDIT and DISP will be in force. Clearly in our situation, the programmer would not normally be around to edit a work session from the keyboard, and furthermore this would undesirably hold up the machine. We therefore decided to make the default options NOEDIT and DISP. This has the added advantage that the user does not have to include the

)CARD NOEDIT

command and throughput is decreased by one card per work session.

The necessary patches on the disk are:
Change word /00AF of sector /02B7 to /A000
Change word /0079 of sector /02CA to /A000

(b) SPECIFICATION OF CARD MODE AT IPL

The standard APL/1130 system requires that, if the first sign on after IPL is to come from cards, then switch 1 of the console entry switches must be up. This means that operators must remember to put up switch 1 when changing to the APL disk, since the 1130 monitor requires all switches to be down at cold start time.

We decided to reverse the meaning of switch 1, i.e. now it is necessary to put switch 1 up if the first sign on after IPL is to come from the keyboard. Thus all switches are down for both normal FORTRAN and card mode APL.

The necessary patch on the disk is:
Change word /0047 of sector /02CA to /4C28

(c) DELAY BEFORE SIGN ON

On the standard APL 1130 system, before a sign on, the carriage returns, waits for about a second, and then spaces six before accepting the sign on. The wait of a second has been removed with the following patch on the disk.
Change word /0036 of sector /02CA to /719C

(d) CARD PUNCHES FOR ⍴ , ⍳ , ∇

These three characters, we feel, are probably the three most common of those which require multi-punching or mnemonics. It seemed to us to be a good idea to try and put these on the keypunch board thus avoiding the use of mnemonics for them (multi-punching is too difficult to teach people who are just beginning computing).

The following patches on the disk achieve this:
On sector /02D1:
Change words /000F,/0010, /0012, /001B and /0025 to /A010, /0060, /2120, /2820 and A040 respectively.
From word /0041 place /D022, /6920, /613E, /C500, /0E8D, F200, /1804, /4C98, /0D6C, /6135, /C500, /0E8D, /F200, /1800, /4C98, /0D6C, /6133, /C500, /0E8D, /F200, /1800, /4C98, /0D6C, /C00B, /6580, /0F13, /4C80, /0D6C.
Change words /0013 and /0014 of sector /02D0 to /4C18 and /0EF1 respectively

The results of these patches are as follows:

i The characters ⍴ , ⍳ , ∇ can be punched as 12-8-7 (upper case T on the IBM 29 Card Punch), 0-8-5 (upper case W) and 8-7 (upper case C) respectively.

ii The)PCH and)PCHS commands will no longer work (the patch from word /0041 of sector /02D1 overwrites most of the routine for punching). However, in the

rare event that someone wishes to punch, the disk can be re-patched quite simply. It is hoped to be able to overcome this difficulty eventually.

iii The standard multi-punches for ⍴ , ⍳ , ∇ will give character errors.

iv The mnemonics for ⍴ ; ⍳ , ∇ will still be accepted.

FURTHER PROJECTS

We have two further projects in mind at present:

(a) In order to make distribution of output to users simpler we will attempt to introduce a "paging" on the typewriter. (This project is under way at present).

(b) In order to speed up through-put we will attempt to implement two new system commands:

i.)PRINT which would be used to switch ouput to the 1132 printer when large quantities of alphanumeric output is required.

ii.)TYPE to switch output back to the console.

We would welcome comments and suggestions for further improvements from people working on a machine under similar conditions to ours.

Mark Sense APL

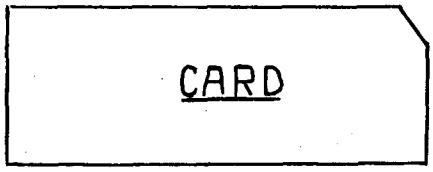
David A. Bonyun
Computing Centre
Acadia University
Wolfville, Nova Scotia

One of the major faults of the IBM 1130 APL system is the requirement that a user either a) sits at the console and hunts and pecks over an unfamiliar and largely unmarked keyboard; or b) keypunches cards using function names for many of the symbols. This inconvenience becomes acute if one tries to teach APL. Traditionally, those learning APL did so from a terminal typewriter and did not have to learn alternative names or tie up a whole computer for long periods of time.

The problem is one to which many people have given some thought. The pages of **Quote-Quad** have held various ideas on the subject and various possible solutions. I offer here another, hopefully better, solution – Mark-Sense APL.

It has been fairly common for school systems and other like institutions to use Mark-Sense cards for training purposes. The obvious advantage is the lack of a required key-punch machine. Mark-Sense FORTRAN, Mark-Sense ICL are both used, as is Mark-Sense BASIC.

Basically, there are two kinds of mark-sensing available. The 360 takes a Mark-Sense reader for which the marks are made vertically on a card. The 514 Reproducing Punch can be equipped with mark-sensing and for this equipment, a horizontal mark is used. Considering the fact that the problem exists for an 1130 to a much greater degree than for the 360, it is not surprising that we have chosen to build our system around the second type of mark-sensing equipment.



This size would have been too small. See inside front cover. -G.H.F.

The card we used is illustrated above. Certain of the system commands are available