MANAGEMENT OF APL TIME-SHARING ACTIVITIES

J. Higgins and A. Kellerman
Computer Center
State University of New York (SUNY)
Binghamton, New York

Introduction

The management of a terminal system at a university or industrial installation provides a formidable task. The user needs take arious forms: (a) an educational program in the syntax of the languge and techniques of programming to take advantage of the attributes of the language, (b) consultation on programming problems (both trivial requests and those involving desing, format, and construction of complex tasks, (c) publicity on operational considerations such as hours of operation, the location and availability of terminals, scheduling, etc., (d) documentation on existing programs and packages, and (e) assistance in administrative activities such as the restoration of working copies of damaged programs, groups, workspaces, etc., and the transportation of packages from our installation to another. The successful management of a terminal system such as APL then involves not only the proper maintenance and honing of the system to insure optimal utilization of computer resources for day by day activities, but well defined procedures for providing the additional personnel support to satisfy the above stated needs.

SUNY-Binghamton's APL System

SUNY-Binghamton has offered APL since the summer of 1967 and currently operates the XM6 version of OS APL/360. There are 1750 APL account numbers on the system; some of these shared by a number of users. Practically every department on campus - from theatre to geology and business to nursing - uses APL, with various emphases. In addition to supporting the local campus, seven sister SUNY institutions and six area high schools access our APL system.

The instruction and education section of the Computer Center offers a variety of forms of education and consultation to [illegible] users. Both potential and experienced users - students, faculty, and staff - feel free to request support at various levels and have received a reasonable degree of satisfaction. Initially, here as elsewhere, potential users were being solicited. Now, rather than acting as apostles and missionaries, we are in a position of resoponding to ever-growing demands for service from existing and potential users. This change in the nature of support is very gratifying, yet presents certiain problems.

Coincident with these rewards of satisfaction, the generation of enthusiasm, and staff motivation are the assorted and varied problems of developing the most effective system for all levels of user education, of effectively motivating and supporting worthwhile classroom and individual projects, of developing ways and eans of evaluating user and system performance, of maintaining the system, and of general administration with limited staff, facilities, and budget. These problems with these restraints are present to some degree in all installations supporting terminal systems. It therefore seems appropriate to present some of our experience, problems, solutions and attempts at solutions with the hope of developing a dialogue with other installations.

It is the purpose of this paper, then, to discuss those problems inherent in maintaining the system and in providing sufficient documentation and "publicity" on the availability of the system and its features, and in posing some partial solutions for providing support for ensuing generations of a core of competent and satisfied users.

Some Approaches

Terminal Allocation. It is an axiom of time sharing that accessibility of terminals to users increases usage to a very large extent. It is therefore desirable to have terminals dispersed in strategically located positions to encourage usage. This provides considerable problems, however. It is desirable to have terminals proctored for various reasons, including programming assistance, terminal maintenance, and general supervision of scheduling and use. These proctors are usually undergraduate students who, in addition to carrying out the above tasks, interface well with students and faculty on a one-to-one basis. Financially it is not possible to proctor locations, which are scattered around campus and house only two or three terminals.

For the most part we have avoided any serious problems by having some diverse locations of terminals periodically checked; two large terminal rooms, containing 22 terminals, are constantly proctored.

System Maintenance. In the spring of last year, 1971, space was rapidly depleting for saving of APL workspaces on two 2314 packs, with no prospect of adding an additional pack. APL users were encouraged to cut down on the amount of material saved and the additional workspace allotment was strictly controlled. However, by the beginning of May, with users getting "NO SPACE" messages, the situation was critical.

The policy at that time of deleting users who have been inactive for three months was not generating space fast enough.

As a last resort, all users were required to turn in a written form, giving their account number and the workspaces that they wanted to be maintained on the system.

After several abortive attempts resulting from misinterpretation of the documentation on the standrard APL IBM utility, lack of complete documentation on the APL utility, our inability to fool the utility into accepting an incremental dump tape for a full dump tape, and various other blunders, the following procedure was implemented. With the help of three programmers and a keypunch operator, cards were punched for each requested workspace. A full dump tape was made and new APL packs were created. From the APL utility, an ACCT 0 to tape was made. This ACCT 0 lists all users and account numbers. This tape was accepted with a program that punched cards with the system command)ADD for each account number on the old system. This deck was read into the 1050 (a terminal equipped with a 1056 card reader) to add all users to build the directories, to the newly created APL packs. The process was very slow, since the 1050 reads a card every 6.7 seconds and there were 1576 cards.

The worskspaces, for which "save forms" were turned in and for which a card had been keypunched, were restored to the new system, using the APL utility which can restore 100 workspaces at a time.

A complete backup set of tapes was kept for people who neglected for various reasons to submit save forms - for later recovery.

Although this method worked sufficiently well, there were several objections to it. First, the very fact that the final procedure was the result of a series of blunders with no better solution in sight left ilittle comfort. Second, the process of keypunching the cards for the workspaces and processing them through the 1056 reader was very time consuming. The paperwork was a nuisance. The retrieval of WS's from the old packs and the creation of the new packs monopolized the computer for a full day.

These disadvantages coupled with the fact that there was a general displeasure in the amount of information conveyed by the form of the APL account number lead to our present method. This method hopefully will be updated to something better, perhaps tied to the addition of files, to improvements in the APL utility, and to broadening the scope of the system commands to handle multiple entries.

Account Numbers. Instead of assigning numbers according to a 5 digit department code with the last four digits of a social security number, for a nine digit number, the following scheme was adopted. The first digit represents status, the next five represent department, and the last three are assigned sequentially according to edepartment. The status digit consists of 1-4 for undergraduates, 7 for graduates, 8 for faculty, and 0 for people who we feel need their number for only one semester. In the past everyone was arbitrarily assigned 1 workspace. Now 0 workspace quota is given to people who are using the CAI packages. The 0 numbers are deleted every semester; the 7 and senior numbers in June. The process for deletion is carried out in the following manner. A general purpose selection program is run against the APL ACCT 0 (under TSO) produced by the utility to search for 0's or 7's or any particular combination desired. This program creates a data set with the selected records. This data set is accessed with another program that punches cards with the appropriate APL system command and user number. These cards are processed through the 1050. The general purpose selection program can also select records of users of such combinations as all senior biology majors going to Corning Community College who have been connected to the system for over 35 hours. A developing interest in the school is the psychology of the time-sharing user. This type of output, along with information obtained from the I-beam readings, provide much information for on-line collection and analysis of data in this realm.

A similar procedure was used, that is punching a deck with the system command ")LOCK under number," to change over to the new numbering scheme. Users were given one month to copy old information into their new number. The old numbers were deleted with a ")DELETE" deck. We have a ")CONTINUE" deck that can periodically be read through the 1050 to clean up the CONTINUE workspaces that users fail to drop. (Figure 2).

SUMMARY OF THE DIFFERENT SPACE SAVING PROCEDURES

Before			After		
Users on System	Tracks	Workspaces	Users on System	Tracks	Workspaces
GRAND RETRIEVE (MAY)					
1748	7399	2641	1748	4850	1576
LOCK-REASSIGN (SEPTEMBER)					
1640	6268	2010	1476	5395	1694
DELETE 0's STATUS NUMBERS (JANUARY, 1972)					
1769	7521	2461	1292	6572	2115
MARCH STATUS (CURRENT)			AFTER DELETING CONTINUES (MAY 9)		
1777	9025	3124	1777	6960	2288

Use of I-Beams in Monitoring System Usage. Using an APL function, MONITOR, requiring a privileged terminal, information can be obtained on a continuous basis on specific port usage, specific account number usage (such as histograms of graduate student usage throughout the day), and total numbers of users in a given time interval. From the data collected and from the results of any desired additional statistical analysis, decisions can be made concerning terminal usage, location suitability, suitability of APL schedule as well as information on amounts of use by different types of users. (Figure 3).

By using I-beams 1-14, which require a privileged terminal, various information about the APL system performance can be collected on line. (Figure 4).

The I-beams, representing histogram data, return a vector of integers seach element of which represents a full word of data. Since this information, collected from the time APL starts running until shutdown, is collected in half-word counters, each I-beam vector has to be decoded, split into its two half-word components with the following APL function. (Figure 5).

For example I2 - represents the system reaction time from when the user's return is detected, until his workspace is dispatched. (Figure 6).

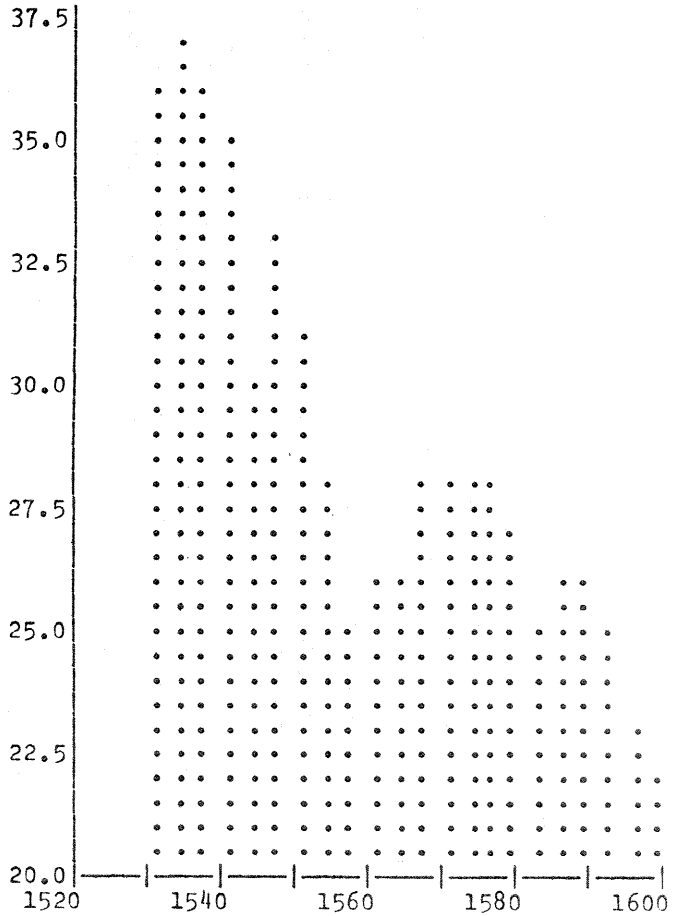
Although our experimentation with the I-beams is still rudimentary, attempts are being made to use this data as input to a simulated time-sharing system, for studying system performance under different loads, and for analyzing the behavior of the time-sharing user.

Priority and Quantum. When operating in a multi-programming environment, the effect - depending on factors such as configuration, number of terminals connected, types of jobs - of APL on batch jobs and vice versa can be substantial. Various parameters internal to APL can be adjusted. APL ensures that other partitions receive frequent CPU service by alternating its own priority between high and low. When APL has a low priority, other partitions will get CPU service. The normal proportion of time that APL has high priority is controlled by a function PRIORITY a,b, which is distributed with the workspace, OPFNS. The priority proportion varies approximately linearly from ---------- depending on the number of ports in use. There are other factors involved such as the quantum, the time allotted an active workspace in core, that can also be set internal to APL by an APL function.

Psychologically a time-sharing user desires at most a 3-5 second response time, (depending on the complexity of the request) but batch users object to at times 400% degradation in their jobs caused by APL. Hence some compromises have to be made. See Figure 7 for comparison data where the priority and quantum have been varied.

Security of the APL System. Theft of numbers of unauthorized users who search wastepaper baskets, unauthorized copies of OPFNS, disastrous experimentation by inquisitive but well-meaning users who desire to probe the mysterious inner workings of APL, and mischievousness make security an annoying but necessary task. As far as the Computer Center has determined no simple procedures or solutions are in evidence in the current IBM APL release. Various attempts at devising elaborate check functions for privileging "authorized" users at terminals outside the confines of the Computer Center have always been cracked.

Beyond these basic considerations there are the very real problems of offering security of creativity to those who desire it. With the possibility of patents for original algorithms and use of functions for trading with other installations or for publishing, workspace and function



NUMBER OF TERMINALS CONNECTED
VS
TIME OF DAY

Figure 3-a

NUMBER OF HOURS CONNECT TIME VS. PORT NUMBER.

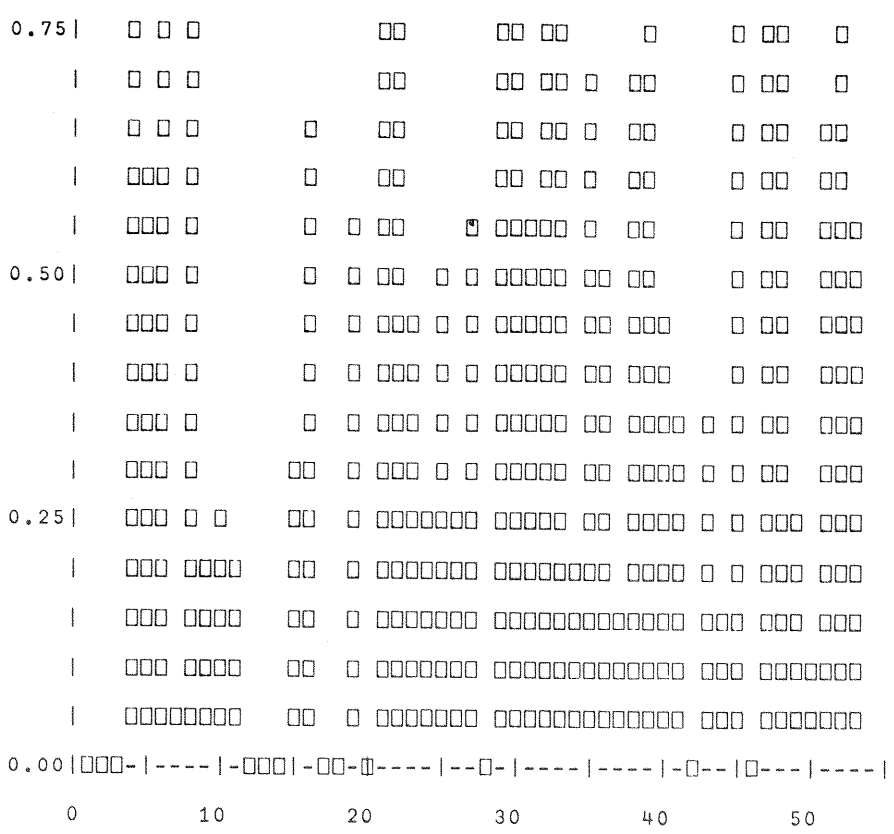


Figure 3-b

Figure 3-c

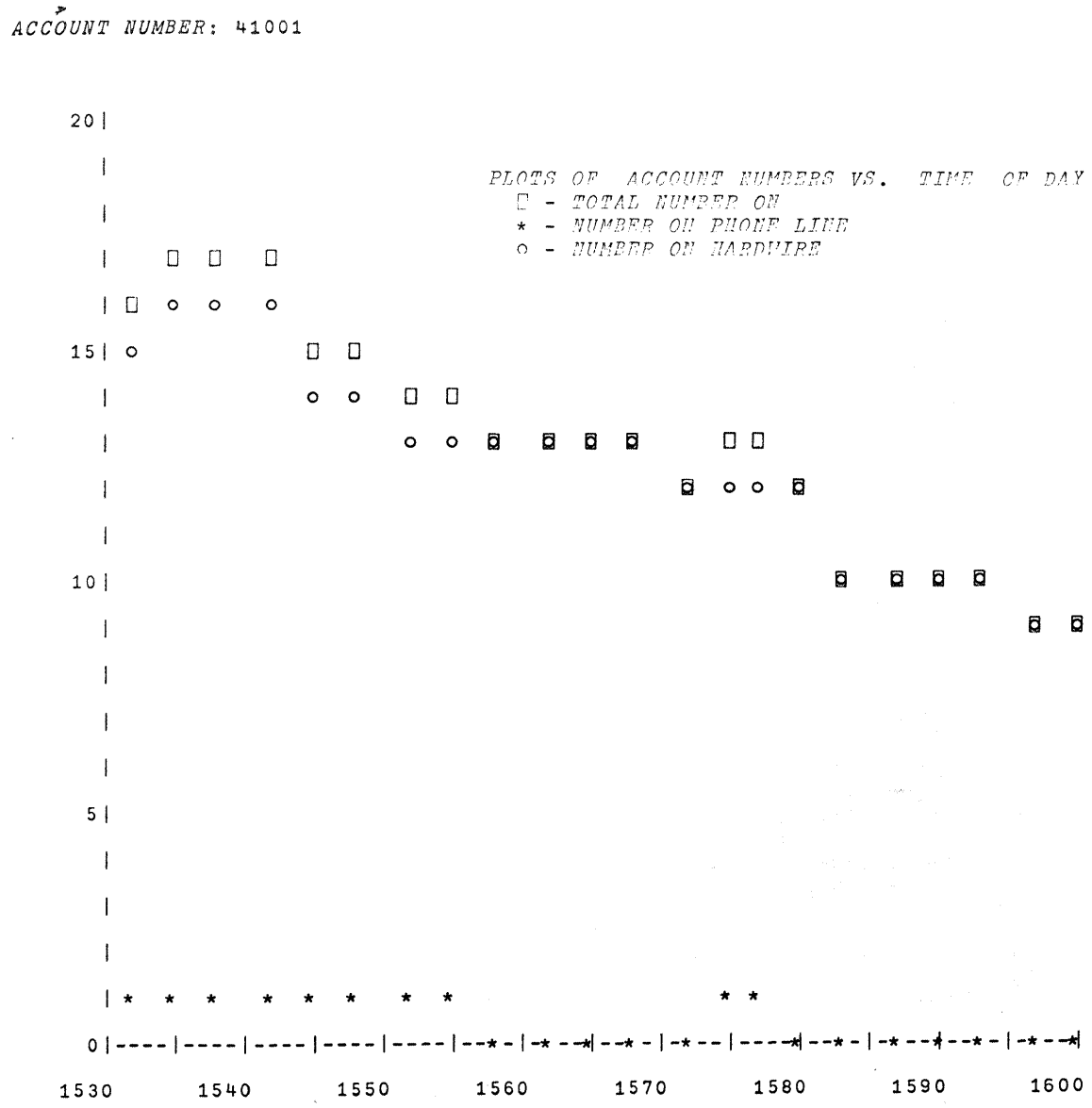

ACCOUNT NUMBER: 41001
PLOTS OF ACCOUNT NUMBERS VS. TIME OF DAY
□ - TOTAL NUMBER ON
* - NUMBER ON PHONE LINE
o - NUMBER ON HARDWIRE
20
15
10
5
0
1530
1540
1550
1560
1570
1580
1590
1600

Figure 3-d

```
      ∇MONITOR[□]∇
    ∇ DELAYΔ MONITOR UNTIL;P;A;CO
[1]   TIMER←TERMNUM←⍳0
[2]   UNTIL← 72 60 ⊥2↑UNTIL
[3]   'INPUT ACCOUNT NUMBERS YOU WISH MONITORED ONE AT A TIME  INPUT STOP
[4]   NUM← 0 5 ⍴0
[5]  L2:→(∧/'STOP'=4↑P←⍞)/EON
[6]   NUM←NUM,[1] 1 5 ⍴ 5↑P
[7]   →L2
[8]  EON:CONNECT←52⍴0
[9]   MAT←((1↑⍴NUM), 0 3)⍴0
[10] LOOP:DELAY DELAYΔ
[11]  TIME← 24 60 ⊥2↑ 72 60 60 60 ⊤⌶20
[12]  TERMNUM←TERMNUM,⌶23
[13]  TIMER←TIMER,100×+/(72 60 ⊤TIME)÷ 1 60
[14]  CONNECT←CONNECT+((⍳52)∈ON)
[15]  A←((1↑⍴NUM), 1 3)⍴CO←0
[16] INL:A[CO;1;]←(⍴P),(+/P<21),+/21≤P←ACCT NUM[CO←CO+1;]
[17]  →(CO<1↑⍴NUM)/INL
[18]  MAT←MAT,[2] A
[19]  →(TIME<UNTIL)/LOOP
[20] ABORT:'NUMBER OF TERMINALS CONNECTED VS. TIME OF DAY.'
[21]  30 90 PLOTT TERMNUM VS TIMER
[22]  10 1 ⍴' '
[23]  CONNECT←(CONNECT×DELAYΔ)÷3600
[24]  'NUMBER OF HOURS CONNECT TIME VS. PORT NUMBER.
[25]  PORT←⍳52
[26]  30 90 PLOTT CONNECT VS PORT
[27]  CO←0
[28]  10 1 ⍴' '
[29]  ' PLOTS OF  ACCOUNT NUMBERS VS.  TIME  OF DAY'
[30]  '   ';PC[1];' - TOTAL NUMBER ON'
[31]  '   ';PC[2];' - NUMBER ON PHONE LINE'
[32]  '   ';PC[3];' - NUMBER ON HARDWIRE'
[33]  3 1 ⍴' '
[34] LP:'  ACCOUNT NUMBER: ';(' '≠NUM[CO;])/NUM[CO←CO+1;]
[35]  →(0≠+/+/+/MAT[CO;;])/(⌶26)+2
[36]  →((⌶26)+2),0⍴□←'NONE OF THIS ACCOUNT NUMBER SIGNED ON'
[37]  30 90 PLOTT MAT[CO;;] VS TIMER
[38]  '

[39]  →(CO<1↑⍴NUM)/LP
    ∇
      ∇ACCT[□]∇
    ∇ R←ACCT N;OR;X
[1]   →((∧/'0'=N),1=⍴N←(' '≠N)/N)/ZER,STAT,0⍴OR←6⌶ 0 0
[2]   R←1↓⌊(7⌶108)÷1000
[3]   R←(10*5)×R-⌊R←R÷10*5
[4]   R←⌊R÷10*5-⍴N
[5]   →ZER-2
[6]  STAT:R←1↓⌊(7⌶108)÷10*8
[7]   R←(X∈R←(R=10⊥'0123456789'⍳N)/ALL)/X←ON
[8]   →ZER+1
[9]  ZER:R←(((1↓7⌶108)>1000000)∧(1↓7⌶108)<100000000)/ALL
[10]  OR←6⌶0,OR
    ∇
```

Note: Good for 52 ports. Additional functions used are found in the Operator's workspace.

Figure 4

I-Beam Number	Unit	No. Of Elements	Max. Value	Significance
0	-	13	-	Count of special disk operations. The elements of the decoded vector give the number of times each of the following system commands has been used: DROP, SAVE, LOAD, COPY, ADD, LIB, OFF, DELETE, LOCK, UNLOCK.
1	1 percent	100	100	The percent of elapsed time given to service an input.
2	1/60 sec.	240	4 sec.	The system reaction time from when the user's return is detected, until his workspace is dispatched.
3	1 second	120	2 min.	User keying time, from the time the keyboard unlocks, until the user hits return.
4	1/60 sec.	120	2 sec.	Compute time per input.
5	(a transfer vector of absolute addresses)			
6	1 minute	120	2 hours	Connect time for each session.
7	1 second	240	24 sec.	CPU time for each session.
8	1 byte	148	148 bytes	Raw input character count, including backspaces, etc.
9	1 second	120	2 minutes	Input arrival time (from one carrier return to the next).
10	1 byte	148	148 bytes	Internal output line length.
13	250 bytes	200	50000 bytes	Garbage in workspace at time of swap write.
14	250 bytes	200	50000 bytes	Active size of workspace at time of swap write.

Figure 5

Decoding the I-Beams

The data in core storage is read-out by a histogram I-beam in increments of full words. To decode the I-beams the following APL function can be used.

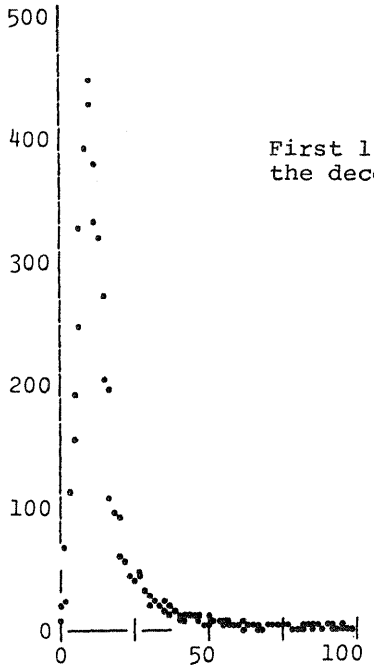
```
     ∇SPLIT[□]∇
   ∇ R←SPLIT I
[1]  R←(1↑I),,⍉ 65536 65536 ⊤1↓I
   ∇
```

Figure 6

```
      IBEAM
THIS WORKSPACE HAS COLLECTED A GRAND TOTAL OF 1 IBEAM READINGS.
THEY ARE FOR THE FOLLOWING DATES AND TIMES:
01)  10:49:19AM ON 02/14/72
WHICH DATE AND TIME DO YOU WISH? (SPECIFY BY NUMBER)
□:
      1
WHICH IBEAM  (0-14 EXCEPT 5) DO YOU WANT?
□:
      2
DO YOU WANT THE UNCODED IBEAM DISPLAYED?
□:
      YES
371202  31064072  1310742  4456558  10158272  16318793  25559486  27984253  21627201  17694924  12845163  6226010
      3997752  3538989  2621478  2818094  1966109  1245207  1441812  1507343  1245197  1048589  458762  458765
      655372  655367  262148  720903  458758  458756  458757  262148  327688  65540  262146  65540  65541  196610
      131074  196611  196611  1  65538  65540  65538  131076  65540  262145  196608  262144  1  65537  0  65536
      131072  65536  196609  196612  65538  65539  65536  65537  3  4  0  196609  0  0  0  2  1  0  3  65536
      2  65536  1  196608  65537  2  131072  65536  65536  131073  65536  1  65536  2  65536  131072  0  0
      0  131072  2  0  65536  65536  65536  65536  65537  0  65537  1  0  1  65538  1  0  2  2  1  0  0  65536
      131072  0  0  0  0  131072  184

  DECODED VECTOR IS:474  8  20  22  68  110  155  192  249  329  390  446  427  381  330  321  270  204  196
      107  95  90  61  56  54  45  40  38  43  46  30  29  19  23  22  20  23  15  19  13  16  13  7  10  7
      13  10  12  10  7  4  4  11  7  7  6  7  4  7  5  4  4  5  8  1  4  4  2  1  4  1  5  3  2  2  2  3  3
      3  3  0  1  1  2  1  4  1  2  2  4  1  4  4  1  3  0  4  0  0  1  1  1  0  0  1  0  2  0  1  0  3  1
      3  4  1  2  1  3  1  0  1  1  0  3  0  4  0  0  3  1  0  0  0  0  0  0  0  2  0  1  0  0  0  3  1  0
      0  2  1  0  0  1  3  0  1  1  0  2  2  0  1  0  1  0  2  1  1  0  0  1  1  0  0  2  1  0  2  0  0  0
      0  0  0  0  2  0  0  2  0  0  1  0  1  0  1  0  1  0  1  1  0  0  1  1  0  1  0  0  0  1  1  2  0  1
      0  0  0  2  0  2  0  1  0  0  0  1  0  2  0  0  0  0  0  0  0  0  0  0  2  0  0  184
 A PLOT OF THIS IBEAM IS AS FOLLOWS:
THE NUMBER OF OVERFLOWS WAS: 184
THE TOTAL OFF-SCALE READINGS WERE: 474
```

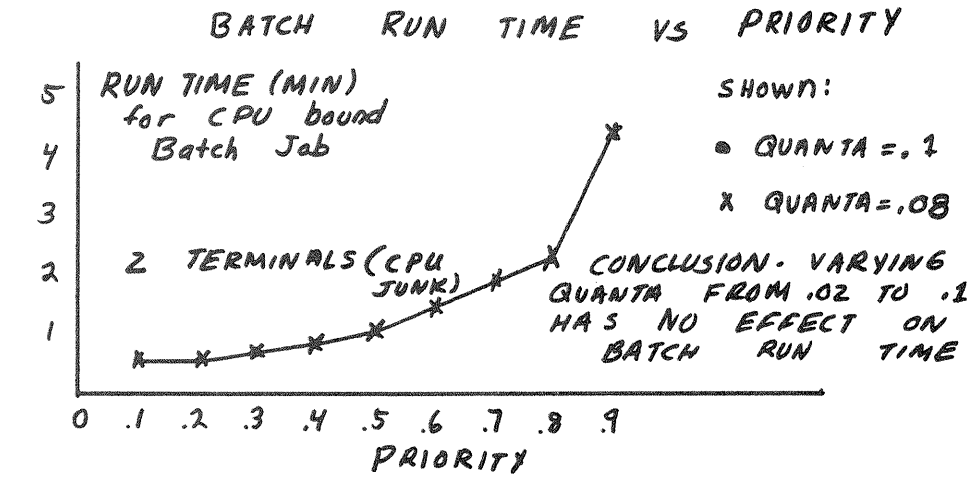
I2: The system reaction time from when the user's return is detected, until his workspace is dispatched.

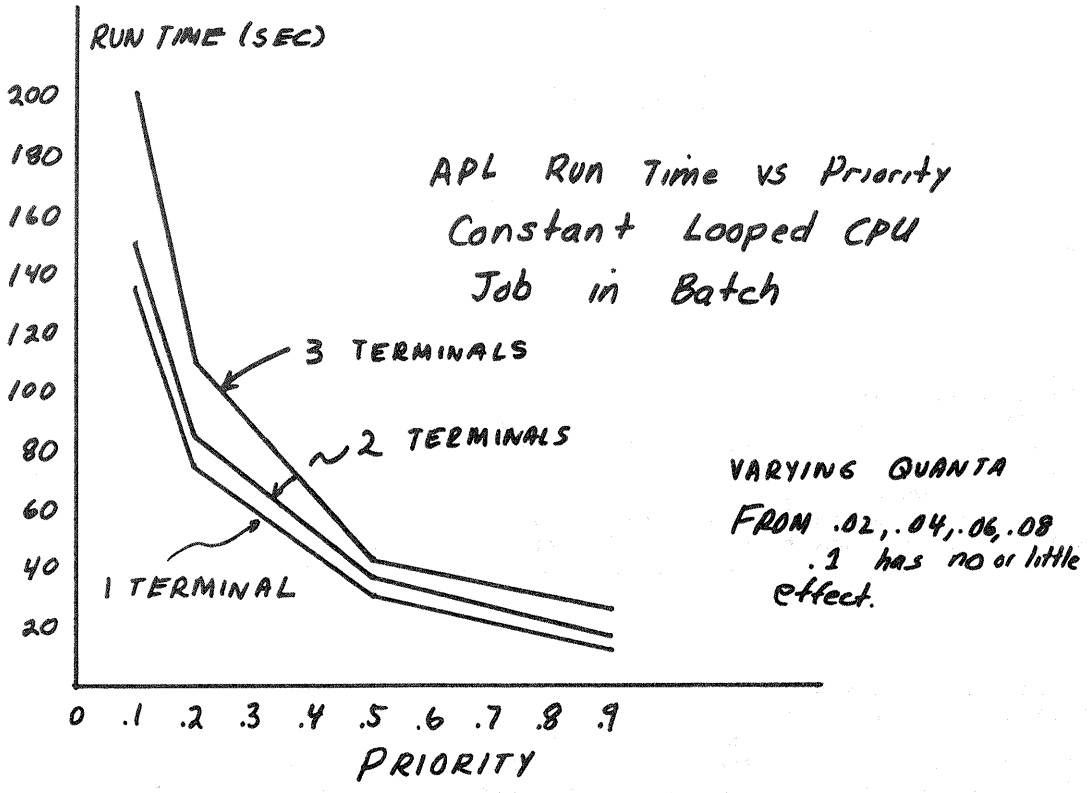


First 100 elements of the decoded I2.

Units of 1/60 of a second

Figure 7


BATCH RUN TIME VS PRIORITY
RUN TIME (MIN) for CPU bound Batch Job
5
4
3
2
1
SHOWN:
• QUANTA = .1
x QUANTA = .08
2 TERMINALS (CPU JUNK)
CONCLUSION - VARYING QUANTA FROM .02 TO .1 HAS NO EFFECT ON BATCH RUN TIME
0 .1 .2 .3 .4 .5 .6 .7 .8 .9
PRIORITY


RUN TIME (SEC)
200
180
160
140
120
100
80
60
40
20
APL Run Time vs Priority
Constant Looped CPU
Job in Batch
3 TERMINALS
2 TERMINALS
1 TERMINAL
VARYING QUANTA
FROM .02, .04, .06, .08
.1 has no or little effect.
0 .1 .2 .3 .4 .5 .6 .7 .8 .9
PRIORITY

security and some form of credit has to be given to programmers and researchers while encouraging APL users to share their work with the world.

Another security problem of a different nature is the disappearance of the contents of workspaces accidentally or because of internal damage. If users report their mishap to the Center within the period of a cycle of dump tapes (3 weeks) their workspace can usually be recovered. We do not publicize this capability.

Occasionally workspaces come out DAMAGED (IMPEACHED) in the twice weekly dump-restore procedure meaning that they are questionable, but are dumped and restored. We try to eliminate the questionability by either copying them over or restoring from a backup tape. Whether this is necessary or not, we do not know. For example, we have discovered that the upper and lowercase idle character cause an impeached workspace, but use of the functions containing these characters is in no way hampered.

To our dismay in March, because of later to be discovered problems in formatting the disks, we had several workspaces DAMAGED and REJECTED as opposed to being DAMAGED and IMPEACHED. The contents of these workspaces are gone. But their name remains in the directory. If a user attempts to save into these workspaces, APL abends. It is necessary to bring APL up again and at all possible speed)DROP these workspaces from the directories.

Education. There is no undergraduate computer science program at SUNY-Binghamton, hence there are no "programming" courses. There are graduate courses (in APL & PL/1) in the School of Advanced Technology which undergraduates can take, by petition, for credit. The vast majority of students, staff and faculty look to the Computer Center for instruction. We have produced a series of video instruction (9 tapes, 45 minutes each) which introduce students to APL/360. The tapes are run, with a knowledgeable Center employee in attendance, twice a semester - usually twice a day (noon and 6:00 p.m.). From two to three hundred people per year are introduced to APL in this manner. The Center employee is necessary primarily to provide the encouragement and motivation for all students to get on the terminal as soon as possible and not just take the video course for theory. During each summer a special class for faculty only is held, and is well attended. There is a 50 page supplementary manual available which is used to follow the video tapes - primarily for the purpose of discouraging note-taking during the tapes. Copies are available.

In addition to the video classes, which are well received, we offer live classes for groups, such as individual classes, on a demand basis. There is also a series of workspaces in APL which instruct a user in the APL syntax in a "CAI" mode. We offer personalized instruction to users who read the User's Manual or our Quick Guide on their own. The Quick Guide is a brief introduction to APL/360 with notes concerning specifically our installation, and sample executions of some of our public libraries. This guide was written to hand to people who stop in the day after the APL classes end and ask when we'll be teaching an APL class. Copies are available on request.

Assistance to Users

In providing assistance to faculty we find various categories of needed support: (1) those who are sophisticated programmers, have good ideas for applications in their courses, and merely request account numbers for their students, reservation of terminals, and perhaps demonstrations of APL in group sessions; (2) those with good plans for applications, but lack ideas for implementing them; (3) and those whose only attribute is enthusiasm. The last two categories of people are best handled on a one-to-one basis, trying to adapt their needs to techniques and existing programs. With a sufficient number of examples of problem-solving techniques, simulation and tutorial programs they can find something consonant with their interests that will provide the supplement or embellishment to their course that they sought.

Students

Students who use the APL system do so also for various reasons: (1) course requirements, (2) their own research or other class work, (3) general unguided curiosity, and mass productions of SNOOPY posters. However, it is true that most students become, for various reasons, much more sophisticated and elegant APL programmers than faculty and that a great deal of course-related APL work can be traced to student initiation by suggestion or actual development.

Computer Center Assistance

In most cases, APL projects have the most success when they are tailored to a specific professor and class. We have, however, developed some general purpose CAI techniques that are applicable to various circumstances. We are currently evaluting the Author Tutorial System

available through IBM. The object is to allow professors, with a limited knowledge of APL, to construct tutorials and drills. Students can respond to questions in free form sentences. Statistics of student performance can be obtained.

In terms of particular applications, some ideas have required a great deal of effort on our part and on the part of the originator. The first step is to determine whether or not the project is "worthy" of implementation.

The determination of the "worthiness" of a given application is not well-defined. Probable use, time, and limited personnel constitute the primary constraints. It frequently goes beyond differentiating what is or is not a good APL application. What we may conceive of as a "bad" application can, in some instances, serve a definite need. Many applications such as the CHEMLAB, APL laboratory monitor, are not cost effective yet, but represent excellent prototypes.

Certain modificaitons of ideas and procedures invariably must occur to make them suitable for APL implementation. Interestingly enough, because of the ability of APL in simulating experiments to their very limit, many of the planned Freshman Physics lab experiments were modified - mainly because the original experimental procedures, taken to the limit, produced less acccurate results than the modified procedures on APL. The necessary algorithms must then be developed, and then coded. Program editing, a continual dialogue between programmer and originator, is an interactive process that can be very time consuming. Once the programming is completded, arrangements are made to allow students easy access to the programs. Student reaction is an important ingredient in the determination of modifications and embellishments.

Documentation

It is axiomatic in user service-oriented organizations that effective publicity is an all-important ingredient for success. We publish the Computer Center Newsletter four or five times a year. (If you'd like to be on our mailing list, we have applications with us.) APL news gets the most coverage. We also publish a list of public library workspaces and their contents; we also have copies here for distribution. We also maintain standardized "on-line" documentaiton.

A large number of our APL users are interested in statistical functions. We have two statistical packages: STATPAK and a package from New Paltz. Several additions have been written by SUNY-Binghamton people. Unfortunately, a large portion of potential users know statistics but cannot understand the descriptions that use a large amount of APL terminology. Our student proctors know APL, but not statistics. We have developed a descriptive workspace STATHELP which gives even additional help to bridge this gap.

We have implemented a MATRIXHELP that describes some things that can be done with matrices and APL and points to other matrix workspaces in the public libraries. A FORMATHELP workspace contains functions and help to format data, functions and help in writing CAI and directions on use of the various plot functions that have accumulated in our libraries.

Future Efforts

Some areas of future emphasis, in addition to those mentioned above, include more concentration in Psychology, the School of Management, the School of Nursing, and applied mathematics in the School of Advanced Technology (SAT).

Since we do not currently have a file system with our APL system we are restrained by the 36K WS limitation - especially for statistical applications, long simulations, and CAI programs requiring the logging of student statistics. We feel a distinct need to provide more information to users on good programming habits and on time/space tradeoffs. A programmer in SAT, Grant Sullivan, has done some investigation in programming techniques to save space and time/space tradeoffs. His work provides some help and guidance in good programming techniques in the above areas.

Of the 1750 users on our system a relatively small proportion exhibit exceptional programming skills. It is a testimony to the efficiency of the APL/360 implmementation that less than good programming does not necessarily punish the user. There are users who are very clever with the APL syntax but do not use it well in everyday practice. There are, of course, users who, no matter what the amount of effort, will never be good programmers. It is probalby impossible and certainly impractical to impose restrictions on the user community to attempt to enforce programming standards. We would, however, like to increase computer-related skills in all areas.

Some of this improvement comes with knowledge of basic computer concepts and numerical methods. Most of our users do not have the time to dedicate several courses to achieve this type

of knowledge. So we are in the poosition of having to consider ways of capsulizing APL and statistics, advanced APL, numerical methods useful in coding APL problems, etc.

Finally, there is the task advising users what system to use. Initially we did not offer any choice of conversational terminal facilities. However, we currently run TSO and anticipate situations, such as taking the determinant of 50 x 50 matrices, where our advice will be to channel the application to the most applicable terminal system.

The ultimate goals are to transport as many useful programs to our system as we can, to provide a large base of available routines, to encourage the development of curriculum materials in our consortium, to adequately publicize that which is available, and to provide the consultation and assistance necessary to eiliminate or reduce impediments to general development. We feel we are in an embryonic stage now, but look forward to increased service to our users. We would appreciate sharing experiences and programs with other installations. There is much to be gained by cooperative efforts.