Designing "Bullet-Proof" Code

EVERYTHING you ever wanted to know about APL Security
...plus 234 additional bonus pages!
NOTICE

regarding phone calls to Kingston, using the IBM internal tie-lines

Effective 01 June 1985, the tie-line phone access for the entire Kingston IBM site will change from "373-" to "695-".

If you call The APL Hotline before 01 June 1985, use T/L 373-1234.

If you call The APL Hotline after 01 June 1985, use T/L 695-1234.

(Outside phone access will not be affected)
Some production notes

The text for this newsletter was entered on an APL-featured 3279 display terminal, and composed using Script. The master pages were then printed on the IBM 4250 electro-erosion printer [a 600-pel matrix printer].

The illustrations scattered throughout the newsletter (including the cover) were created between 1850 and 1925, originally appearing in newspapers, advertisements, and periodicals. The particularly astute amongst you may notice that many of the old engravings have been slightly retouched (using a 5¢ Gem razor blade and some Scotch tape).

Cover Portrait: Everyone tells you to write “bullet-proof code”, but no one seems to tell you how to do it. We will.
"What Ever Happened to the Jot-Dot Times?"

We hear that question all the time. Well, first of all, we should point out that we are still on schedule. Realize that every issue of Jot has stated that it is "published sporadically," and we certainly don't want to break with tradition now.

For those of you who may consider that answer to be a bit too flippant (and who wouldn't?), we'll offer another defense. We would dearly love to be able to publish the Jot-Dot Times on a frequent basis. And, no—contrary to some opinions which have been voiced—we haven't been this long because we didn't have anything to say (...those of you who know us, know better). What we have been short of is time and manpower... but then, who isn't?

One of our projects which took a big chunk of time was the publishing of a user's guide for the new APL2 Program Product. If you are interested in such things, you can order "An Introduction to APL2" (SH20-9229) from the Mechanicsburg or Copenhagen distribution centres. That was our interim Jot-Dot Times. ...So you see, we really have published another issue—old engravings and all.

Another delay in getting material published was simply the difficulty in dealing with the mechanics of printing the final camera-ready copy for the manuals. All of the previous issues were done on—believe it or not—a Selectric terminal [remember those?], but they are slow and aged... and in a much lesser state of health than they were years ago. We sorely needed better production methods and tools, while still being able to maintain a high calibre of print quality. We think that we have finally found the right approach with the IBM 4250 printer, which was used to print this manual. [For those of you who have never heard of that printer, it's a matrix printer with lots of dots.] We hope that you will like our new look. [Gee, this is just like the big companies do it!]

"You complain, friend Swift, of the length of my epigrams, but you yourself write nothing. Yours are shorter."

—Epigrams I, 110
by Marcus Valerius Martialis, A.D. c 40 – c 104
Balking at Bulk

Realize that the history of the Joto-Dot Times has been to make each new issue larger than all previous issues combined (not really intentionally; it just happens that way). For better or for worse, we once again came frighteningly close to that pattern.

Realize also that the production of The APL Joto-Dot Times has been handled by a staff of one for as long as it has been published (since 1977). And unfortunately, other projects often (usually) have to come first, to the exclusion of our newsletter. For example, we had to meet the Corporate Security requirements.

The result of all of that is that one of the most frequently-asked questions about our newsletter is, “How come I didn’t get the last issue of The Joto-Dot Times? Am I still on your mailing list?” Well, you can see now that you evidently are still on our mailing list. The reason that you haven’t gotten a recent copy is that we haven’t published one since (gulp!) the Summer of 1981 [the Mailbox issue]. Another common question is, “How can I get all of the back issues of Joto?” Unfortunately, we can’t be of much assistance here; we have few remaining back issues of Joto-Dot Times.

Presenting the All-Time Least-Read Issue of the Joto-Dot Times

Okay, we know that Security isn’t necessarily your favorite subject. But we (as the suppliers of service) have a requirement to show you the ins-and-outs of the various security items on APL. Corporate Security Instruction #104A (the definitive word on IBM Security), states, “Suppliers of services must effectively communicate to owners and users available control facilities, recommended practices, and installation rules and restrictions” [page 155]. If we’re going to do that at all, we’re going to do it right.

Finally, while people often balk at the idea of reading about Security, a much worse position to be in is to be required to comply with rules that you have never seen... and further, to not have any information on how to implement the required security measures.

We value your thoughts on this newsletter. If you have any comments or suggestions regarding either the newsletter or our service, please let us know. There is a Reader’s Comment Card folded inside the back cover of this newsletter.

Your comments can often help to mold future issues and future offerings on APL. Many of the facilities that we have offered in the past are a direct result of comments that were sent in by newsletter readers. Please, let’s hear from you!

—The Kingston APL Support Group

1 We are.
# Table of Contents

**Preface** .......................................................... ii

"What Ever Happened to the Jot-Dot Times?" .................. ii

**Introduction** ..................................................... 1

How To Use This Newsletter ................................. 1

**Section 1: Keeping Current** .................................. 5

Using workspace 1 NEWS .................................. 5
Analyzing the Security of the APL Mailbox .......... 6
Security Check-list ....................................... 9
System Backup Procedures for your Data ........ 10
Disaster Recovery Planning ......................... 12

**Section 2: Periodic Mailings and Responsibilities** ...... 13

Quarterly APL Utilization Reports ..................... 14
Bi-Annual User Re-Validation Audit ............. 16

**Section 3: Protecting Your System Sign-on** ........... 19

Telephone Access to APL ............................... 19
Signing On .............................................. 21
Understanding Sign-on Error Messages .......... 23
One-Minute Sign-On Time-Out .................... 27
Logging of Unsuccessful Sign-On Attempts .... 27
"Last-Used Date" and "Password Expiration Date" ..... 27
Controlling Your Sign-On Password ............. 28
Proper Use of Passwords ......................... 31
"Display Inhibit" for Hiding Sensitive Inputs .... 34

**Section 4: Protecting Your APL Workspaces** ........... 37

Workspaces and Libraries ......................... 37
Libraries of Saved Workspaces ................ 39
Locking Your Workspaces ....................... 39
Extra Protection for the CONTINUE Workspace ...... 41
Restricted Libraries (Load-Only Workspaces) ... 42
Confined Accounts (for Non-IBM Access) ....... 43
Introduction

Security: It’s not just for break-ins anymore

The term “security” is generally understood to apply to methods of controlling access to information. In a broader sense, it must also include methods of preserving the integrity of data and protecting resources. Additionally, normal on-going business management control – knowing where your money is going – is a consideration in the use of any system.

Many features of the APLSV system are provided to preserve the integrity of your work and data. These are automatically invoked by the system. For applications which require a higher level of security, other features are available which are invoked by following specified procedures. This booklet discusses both system and application security features and their uses. We provide this information in the hope that you will be able to provide users of your applications with security measures which would be seen as being very secure by an informed user.

Although this booklet is intended to be comprehensive, it is not exhaustive. For very sensitive applications, you may want to consider additional security steps beyond the items shown here. We are always available for discussions of such matters.

Here Today; Here Tomorrow

Security is an ever-present issue. It probably comes as no news to you that Corporate Security tightens the requirements for controlling data on time-sharing systems a bit more each year. This booklet has three main purposes:

- To detail the security requirements which Corporate Security requires you to meet,
- To explain the security features which we provide for your use here on Kingston APL, and
- To explain the steps that we take for you, to help to protect your data.

3 ...other than physically, perhaps.
For the NON-Programmers Amongst You

We often hear comments from our various customers, saying that “I’m not a programmer; I’m not even sure how to spell ‘APL’!” Don’t worry; you’re in good company. We realize that most of the users of the Kingston APL system are not programmers. Our best estimate is that only about 15% of you do any APL programming at all. The first three sections of this booklet give general information regarding the security requirements that all of the users are required (by Corporate) to meet. Additionally, if you have any TSIO datasets on the system, a later section explains in detail each of the security requirements that come with that territory.

For the Programmers Amongst You

The remainder of this booklet—which is the greater portion of it—deals with all of the details of programming which can separate the somewhat secure applications from the very secure ones. Lots of detail is given so that you can use the portions that you need as a reference manual. For those portions which discuss programming details, we of course assume that you are already an APL programmer. If you are not already a programmer, this booklet will not teach you programming. That’s not our intent here. If the details of operations that are shown here make you realize that you need some background before you can make use of this manual, we recommend that you refer to “The APLSV Version 3 User’s Guide (SH20-9087)” and “The APL Language Manual (GC26-3847)”.

At the end of this booklet are complete copies of two important Corporate Security documents. The first one is Corporate Security Instruction #104A (form number ZE22-7020). The other one is Information Classification and Control (form number ZZ00-0001). Additional copies of either of these documents may be ordered from the Mechanicsburg and Copenhagen Distribution Centres, in the same way that you would order any IBM manual. Both are also available on-line in workspace “10 DOC104”. Entries which appear in italics in the index refer to those documents.

Here Come the Excuses....

We’ve heard them all. “Security isn’t my job.” “It’s tedious and boring to discuss that stuff.” “Security is too time-consuming.” “Those documents are filled with crazy requirements.” ...Well, when’s the last time that you read—no, really read—the Corporate security requirements? The early requirements, many years ago, were attacked for not being relevant to interactive systems (such as APL) and for being “un-meetable.” So Corporate took these complaints to heart, and the new requirements are both relevant and reasonable. And they will be enforced by Corporate.

Don’t Shoot the Messenger!!

Corporate Security Instruction #104A requires that “Suppliers of services must effectively communicate to owners and users available control facilities, recommended practices, and installation rules and restrictions” [page 155]. As the suppliers of service for Kingston APL, we are required to both advertise and
meet these security requirements. If you have disagreements with the various requirements, please at least realize that we didn't invent the rules, and perhaps take solace in the understanding that we have had to meet all of the rules, too.

Who Ya Gonna Call??

If you have any questions regarding your use of Kingston APL, please contact:

Kingston APL Support
63C 385, Neighborhood Road, Kingston, NY 12401
☎️ T/L 695-1234 or 914-385-1234 📞
IBM internal systems must be used for IBM management-approved purposes only.

"IBM Confidential Restricted" is the highest security level supported by Kingston APL.
Section 1: Keeping Current

Using workspace 1 NEWS

This document is our attempt to explain all of the current security features and requirements—but “current” is a key word here. Requirements change; all we can do is pass the word along. And new features are added periodically, but they are of little use if you don’t know about them.

We of course always have the option of mailing memos and documents (like this) to all of our users. But with thousands of Kingston APL users world-wide, that’s slow and expensive. For keeping up-to-date with changes as they occur, our prime means of contact with you is through 1 NEWS.

Unfortunately, some of our users never look at 1 NEWS. When time-saving features are developed, they are the last to know. And when new security requirements are passed down to us from Corporate, they may be sufficiently late in hearing about them that they are in “catch-up” mode.

Workspace 1 NEWS is probably the only really universal APL workspace. It is installed on nearly all APL systems (from all companies). While it may take on quite different forms on different systems, it is still the recommended means for hearing the latest news about your system. On Kingston APL, we have set it up so that it will only take a few seconds to check it if there doesn’t happen to be any news, and just a little bit longer if there is some news [...how fast can you read?]. And once you have seen a news item, you won’t be prompted for receiving it again. But all of this is for naught if you don’t ever check the news.

We strongly recommend that you

)LOAD 1 NEWS

each day that you use APL.
Analyzing the Security of the APL Mailbox

Sometimes it's instructive to analyze a sample application to see the types of approaches that may be used. The APL Mailbox (in workspace 1 NEWS) is an application with what we consider to be rather tight security.

Sending Your Own Messages

As well as providing the function of delivering our news messages to you (as its name suggests), workspace 1 NEWS also provides a means of letting you send your own messages to anyone else on the system. We realized some time back that a news facility is really just sort of the junk-mail equivalent of a proper mail facility— it's the portion that delivers the mail to "Occupant". So we generalized it. It is now a full-function mail facility, with provision for setting up distribution lists, specifying release dates and expiration dates, using "carbon-copy" notations, and pretty much whatever else you need to communicate with the other Kingston APLSV users. It's particularly helpful when the person that you're trying to contact is in a different time zone or overseas.

To send a message to another user, just type "SEND", and you will be prompted through the necessary steps. At any point in the Mailbox where you need some assistance, just type a question-mark. There is quite a bit of "help-text" built in.

Along With the Extra Capabilities Comes Concern

When we started to offer these increased capabilities, we of course had a lot of concern about being sure that mail couldn't do astray. There should be no way that any user can print any message except the ones that he created and the ones that have been sent to him. If you were ever to be given the feeling that someone was monitoring your mail, chances are you would never feel comfortable with it, so we wanted to do everything that we could to ensure complete privacy for the mail.

During the design of 1 NEWS, we spent quite a bit of time reviewing security problems and solutions. The more we looked, the more potential loop-holes we found. That's partly what led to this newsletter. So, let's just review the steps that we took for ensuring the confidentiality of the Mailbox application, and see if any of these points may interest you. All of them are covered in detail in this manual.

1. Obviously, the Mailbox itself has to ensure that you can't print anyone else's messages just by using the commands in the Mailbox. Therefore,

   a. \[AT\] is read by the Mailbox to know who you are. This is then correlated to your "mailcode".

   b. A copy of your mailcode is stored in the message header, as a double-check when the message is retrieved. Pointers are used to locate your messages quickly and efficiently, but before you are given a chance to see any message, the system checks to ensure that your mailcode appears in either the "to" or "from" lists. If there's any discrepency,
that would indicate a bug in our code, so the pertinent
details of the problem would be logged in a dataset for us
to investigate further.

2. All of the functions in the Mailbox are locked.

3. Local variables and functions are used throughout the code
(using a function file). In this manner, some of the critical
functions aren’t even in the workspace, so it’s that much
harder to tamper with them.

4. Execution of the code is initiated by the latent expression
($DLX), and the code cannot be re-started manually. When the
workspace is loaded, it resets the latent expression and re-as­
signs another global variable.

If you interrupt the latent expression before it has been
executed and changed, the other global variable hasn’t yet
been set, so the code can’t be re-started. If you execute the
latent expression and wait until the global variable has been
set, the latent expression is no longer valid.

These checks of course don’t constitute a very secure
procedure by themselves, but they help, and since the user
wouldn’t know what the global variable is, or what its value
is supposed to be—or even that such checks are taking place—it’s one more bit of protection.

5. The OPEN function looks at the value of $DLX and the global
variable to see if they are set to appropriate values. If they
are not set properly, the possible attempted mis-use is logged.

a. That OPEN function is one single stand-alone locked
function, so it’s not possible to see what it is doing inter­
nally.

b. If the code determines that a user has loaded the
workspace, interrupted the execution, and tried to tamper
with the OPEN function, it logs the problem instead of
opening the mail file.

c. The OPEN function also looks at the values of several other
variables in the workspace—which should be local to its
calling function—to ensure that it is being called from
within the proper function.

6. The workspace is stored in a restricted library, so that no one
can save a copy of the workspace in their own account and
have it sent to another system to examine the code. This is
one of the most important security steps of this facility.

7. The mail is stored in a reserved dataset, so users must go
through a command dataset in order to be authorized to
access it.

8. The command dataset issues an IRW command with “universal
access” (anyone can use it). Therefore, to further protect
that, pseudo-passwords are used through symbolic parameters
in the indirect commands.
9. The Mailbox’s dataset is protected from batch access by RACF—as are all of the TSIO datasets on the system, by default, unless you explicitly take steps to over-ride it.

10. All of the functions in the Mailbox are protected by several levels of dyadic execute, to protect any messages that you may be in the process of entering, just in case you press **ATTENTION** (or **PA2**), or in case there’s a bug in the code. [Is that possible??]

11. The unlocked maintenance copies of the functions for the Mailbox are kept in a locked workspace.
   a. That workspace is protected by a non-trivial password.
   b. That password is changed periodically.

12. The text of any Mailbox message that you send is scrambled when the message is stored on the file. This is the final protection, just in case someone manages to sneak a peek at the file. Therefore, simply getting access to the file by any means other than through the functions that have been provided in 1 NEWS, will tend to be fairly useless.

   *

Throughout the rest of this newsletter, we will be discussing these points and many others, to demonstrate exactly how to protect the applications and data that you consider to be sensitive.  ■
Security Check-list

Here is a list of items which you should check to ensure the security of your applications. Rate yourself.

<table>
<thead>
<tr>
<th>Have you:</th>
<th>Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Changed your sign-on password regularly?</td>
<td>passwords</td>
<td>28</td>
</tr>
<tr>
<td>□ Chosen a non-trivial sign-on password?</td>
<td>passwords</td>
<td>31</td>
</tr>
<tr>
<td>□ Locked workspaces having sensitive data?</td>
<td>workspace locks</td>
<td>39</td>
</tr>
<tr>
<td>□ Checked to see what datasets you own?</td>
<td>workspace 1 FILELIB</td>
<td>73</td>
</tr>
<tr>
<td>□ Classified all of your datasets?</td>
<td>CLASSIFY function</td>
<td>79</td>
</tr>
<tr>
<td>□ Used reserved datasets for confidential data?</td>
<td>dataset privacy</td>
<td>85</td>
</tr>
<tr>
<td>□ Reviewed authorization lists in IC datasets?</td>
<td>IC datasets</td>
<td>92</td>
</tr>
</tbody>
</table>

Realize that a good score here doesn’t necessarily mean that there’s nothing else to consider (...if that was the case, this would have been a one-page booklet). But this should serve as a reminder of the major points.
System Backup Procedures for Your Data

When you "\(\text{SAVE}\)" your workspaces and write your data to TSIO datasets, it is presumably permanently stored. But everyone knows that hardware isn't infallible; how do we ensure that this information is protected and there when you need it, even if the hardware fails? ...We ensure it by planning ahead and by taking steps every day to recover from the problems which we all hope will never occur (...with luck, this will be "wasted" effort!).

**APLSV Workspaces**

APLSV shuts down each night for workspace and file maintenance, typically from midnight to 2:00 am on Tuesdays through Fridays. During this nightly maintenance, copies of all of the workspaces that were "\(\text{SAVE}\)"ed during the previous day are written out to tape. This is an "Incremental dump" of just those workspaces which have been changed. These tapes are saved for two weeks, and then re-used.

Every Saturday morning, from 2:00 am to 7:00 am, a "Full dump" is run, in which all of the workspaces on the system are copied onto tape, whether they were "\(\text{SAVE}\)"ed during the previous week or not. These tapes are kept for twelve weeks before they are re-used. After they are two weeks old they are sent off-site, and kept off-site for four weeks.

Finally, four times a year we run our Quarterly Backup, which copies all of the workspaces onto tape. These quarterly tapes are kept for two years before they are re-used. All of the quarterly tapes are kept on-site.

Notice that the Password Change chart on page 30 points out that APL accounts are only actually deleted at the end of a quarter. That schedule is based upon this backup scheme, wherein workspaces may be recovered up to two years after the account has been deleted. Notice also that the chart shows that the account is locked for some period of time preceding the deletion, which prevents the workspaces from being used or updated before the account is deleted.

**TSIO Datasets**

Since Kingston APL is a highly interactive system with many millions of updates to the data occurring daily (literally), our intent is to copy all of the TSIO datasets onto tape every night. With the volume of data to copy and the amount of other work to be done each night, this desired goal cannot always be met. If you consider your application to be critical, and need to guarantee daily backup for it, please let us know of your needs.

---

3 Kingston APL typically runs about 22 hours a day, seven days a week. This schedule is an approximation of our normal schedule for discussion purposes. Exact schedules—including any intermittent schedule exceptions—are always available and kept up-to-date on APL... "\(\text{LOAD 1 NEWS}\)" and type "SCHEDULE".
Every Monday through Thursday, datasets are copied to tape during the evening; these tapes are kept for one week. Each Friday the datasets are copied to tape and kept for four weeks, with the newest and oldest being kept on-site, and the middle two weeks being kept off-site.

On-Site versus Off-Site Storage

We have been discussing the idea of sending tapes off-site for storage; perhaps we should discuss why we do that. We would like to keep everything on-site, right in our own tape libraries, so that it’s available for you if you need to have some data recovered. But what happens if there is a disaster, such as perhaps a fire or explosion, which damages or destroys our tape library?

To plan for such contingencies, we regularly send copies of workspaces and datasets to an off-site underground storage facility, whose entire function is to store and protect data against the most dire of circumstances—whatever that may be. The entire point of all of this is to make sure that your data is available for your use, come what may.

Recovering Data from Underground Storage

Backing up all of that data would be an exercise in futility if we didn’t have a reasonable means for you to get it back. Apart from natural disasters, fires in the machine room, and even our own hardware failures, there are cases that occur all the time which put that backup data to use. It’s certainly not unheard of for people to accidentally delete or accidentally overwrite their own datasets. If this happens to you, don’t panic! Just give us a call at T/L 695-1234, let us know whether it is a workspace or a TSIO dataset that you need to have recovered, and give us its account number and name. As long as it has been there overnight, it should be recoverable.
Disaster Recovery Planning

Kingston APL has made arrangements to establish service at another IBM site in the event of a large-scale disaster that would prevent the Kingston site from being used. Tests of this operation, simulating actual disaster recovery, are conducted once a year.

Workspaces would be recovered from the most recent “Full dump” tapes.

It should be understood that we cannot necessarily restore all of the TSIO datasets on another machine following an actual disaster, due to the massive quantities of storage involved. This would of course impact some users since all datasets would not be available on line. Some of the users would have to wait until more permanent arrangements were made. In that event, any decisions as to which datasets or applications would be available would be based upon business needs at the time of the recovery.

Realize, though, that we are discussing here the recovery from a major disaster—a large-scale fire, explosion, tornado, and so forth—of a magnitude that would prevent us from continuing operations in Kingston. Obviously the main direction in recovery is to get operations restored at the Kingston site as quickly as possible.

“...Whom unmerciful Disaster
Followed fast
And followed faster....”

Edgar Allan Poe
The Raven [1845]
Section 2: Periodic Mailings and Responsibilities

Kingston APL sends reports out periodically to those of you who have accounts (sign-on numbers) on the system and to those of you who are paying for the service. Some of these reports are for information only; some of them require a reply. This section will explain what it is that we mail, and why.
Quarterly APL Utilization Reports

Four times per year (at the end of each quarter), we mail an APL Utilization Report to the billing manager of each user on the system. This report has three main purposes: to show the manager who is accessing our system under his problem number, how much usage they have accumulated (for asset control, budget, and planning purposes), and which users have not classified their files, per Corporate Instruction 104A.

The manager who receives the report should ensure that all of the users shown are still in his group. If there is any error in the administrative information that we show, please notify us promptly, using the change form that we include at the end of that report.

Ideally, if the reports can be distributed to the appropriate APL users after reviewing the data, it may help them to track their own usage.

These reports are sent for information only, and do not require any action on your part when you receive them; no reply is needed. However, please do not confuse this with the bi-annual audit forms that we send out; those do require a response. ...We'll discuss those next.
Here is a sample of the Quarterly Utilization Report:

**PROBLEM NUMBER 123XAK**

BILLED TO: J DOE, 123 003-1, ATLANTA, GA (RECD) 8-555-2468
123456 MJSMITH J SMITH, 123 003-1, ATLANTA, GA (RECD) 8-555-1357

<table>
<thead>
<tr>
<th>Dates</th>
<th>Connect</th>
<th>Compute</th>
<th>Tracks</th>
<th>Cost</th>
<th>Compute</th>
<th>Storage</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/29</td>
<td>5.25.00</td>
<td>0.03.31</td>
<td>3</td>
<td>$23.56</td>
<td>$39.04</td>
<td>$0.03</td>
<td>$62.63</td>
</tr>
<tr>
<td>07/06</td>
<td>3.44.00</td>
<td>0.02.15</td>
<td>3</td>
<td>$16.24</td>
<td>$24.98</td>
<td>$0.03</td>
<td>$41.25</td>
</tr>
<tr>
<td>07/13</td>
<td>0.36.00</td>
<td>0.00.25</td>
<td>2</td>
<td>$2.61</td>
<td>$4.63</td>
<td>$0.02</td>
<td>$7.26</td>
</tr>
<tr>
<td>07/20</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07/27</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08/03</td>
<td>7.30.00</td>
<td>0.04.30</td>
<td>3</td>
<td>$32.63</td>
<td>$49.95</td>
<td>$0.03</td>
<td>$82.61</td>
</tr>
<tr>
<td>08/10</td>
<td>3.12.00</td>
<td>0.01.48</td>
<td>3</td>
<td>$13.92</td>
<td>$19.98</td>
<td>$0.03</td>
<td>$33.93</td>
</tr>
<tr>
<td>08/17</td>
<td>4.12.00</td>
<td>0.00.16</td>
<td>4</td>
<td>$6.38</td>
<td>$14.06</td>
<td>$0.02</td>
<td>$20.48</td>
</tr>
<tr>
<td>08/24</td>
<td>4.13.00</td>
<td>0.02.52</td>
<td>3</td>
<td>$18.34</td>
<td>$31.82</td>
<td>$0.03</td>
<td>$50.19</td>
</tr>
<tr>
<td>08/31</td>
<td>4.24.00</td>
<td>0.03.12</td>
<td>3</td>
<td>$19.14</td>
<td>$35.52</td>
<td>$0.03</td>
<td>$54.69</td>
</tr>
<tr>
<td>09/07</td>
<td>1.56.00</td>
<td>0.01.56</td>
<td>3</td>
<td>$8.41</td>
<td>$21.46</td>
<td>$0.03</td>
<td>$29.90</td>
</tr>
<tr>
<td>09/14</td>
<td>3.28.00</td>
<td>0.02.36</td>
<td>2</td>
<td>$15.08</td>
<td>$28.86</td>
<td>$0.02</td>
<td>$43.96</td>
</tr>
<tr>
<td>09/21</td>
<td>3.20.00</td>
<td>0.02.02</td>
<td>2</td>
<td>$14.50</td>
<td>$22.57</td>
<td>$0.02</td>
<td>$37.09</td>
</tr>
</tbody>
</table>

---

[**YTD**] 139.28.00 1.22.52  $576.25 $919.85 $1.19 $1,497.29

Sign-on password last changed 09/21/84; it MUST be changed by 11/23/84

---

**PAST YEAR AT A GLANCE**

LEGEND:
- ☐ Connect hours
- ✔ Compute minutes

[Prepared and printed on APLSV] 123XAK
Bi-Annual User Re-Validation Audit

In operating a world-wide service like Kingston APL, we of course need to know who our users are, and to be able to get in touch with them; good business practices dictate that we keep this information current.

Furthermore, Corporate Security Instruction #104A requires that “Authorization to use IBM’s information processing facilities and services must be revalidated periodically and deactivated upon notice of termination of the user’s business need” [page 157].

Twice a year (during February and August), we send out a User Re-Validation Audit form to each user, which needs to be reviewed, have changes to mailing address marked, be signed by the user and his manager, and returned to us.

When this report is sent out, you will generally have two months in which to return it. We provide such a long time to try to cover all of the special situations that may arise, such as a user being on an extended vacation, or a user in Japan with a manager in Germany.

We strongly recommend that you return these forms promptly when you receive them. Since it is a Corporate requirement that we audit our users, we must enforce the return of the forms. The only means we have of enforcement is to lock offenders out of APL if the forms aren’t returned within the allotted time. We don’t like to take this step—we realize that we are simply a service group and that you need to get your job done—but that’s our only means of enforcing the Corporate rules.
Here is a sample of the Bi-Annual Audit form:

<table>
<thead>
<tr>
<th>Account Number/Name: 123456 MJSKITH</th>
<th>Billed to Problem-Number 123XAK</th>
</tr>
</thead>
</table>

- **NEW PROBLEM NUMBER**
- **OLD PROBLEM NUMBER**
- **FILE COPY**
- **DIRECTORY**
- **SYSTEM (ADMINLOCK)**
- **PROFILE**

Return to: IBM Corporation, APL Administration, 63WD 220, Kingston, NY 12401 (TIL 695-1234)

If form is NOT returned, user will be DENIED ACCESS to Kingston APLSV
Return REQUIRED no later than October 1, 1984

- Check here if this account is to be DELETED from Kingston APLSV
  (Caution: Only if the account is no longer in use)
- Check here if all information is correct
  (Signatures still required)

---

**User’s Information:**

- First and Last Name: JAY SMITH
- IBM Zipcode: 123 003-1
- Mailing Address: ATLANTA, GA
- IBM INTERNAL MAIL
- Division: 26 DSD
- Tie-line and extension: 695-1357
- Employee’s serial number: 123456
- Employee’s signature: Date:

---

**Manager To Be Billed:**

- First and Last Name: JOHN DOE
- IBM Zipcode: 123 003-1
- Mailing Address: ATLANTA, GA
- IBM INTERNAL MAIL
- Division: 26 DSD
- Tie-line and extension: 695-2468
- Manager’s serial number: 112233
- Manager’s signature: Date:
Section 3: Protecting Your System Sign-On

Telephone Access to APL

If you are one of the twenty-percent of our users who use dial-up terminals instead of 3270 display terminals, you will need to understand how we control the telephone access into our system.

Kingston APL has local phone numbers available at many of the IBM locations which use the system heavily. Our system may be accessed through dial-up terminals from all over the world.

Where do you find the phone numbers?

When a new account number is first established for you by APL Administration, they send a memo to your manager verifying that the account has been set up (after all, he’s the one who has to pay for it). Included in this memo is a list of phone numbers that you are authorized to use for dialing in to the system. Those phone numbers are different for each IBM location; that is, if you are in San Jose, you will use a different phone number for dialing in to Kingston APL than a user in Raleigh would use for dialing in to the same system. In general, we try to provide each of the major locations with local phone numbers for accessing our system. Those phone numbers are IBM Confidential (which is why they aren’t provided here).

You are typically authorized to use only the phone numbers which have been set up for your own location. To get a list of the ones that you are authorized to use, ”LOAD 1 PHONES”. Due to changes in local requirements, phone numbers do tend to change periodically. To keep current with those changes, we recommend that you ”LOAD 1 PHONES” and print out the list every month or two. In that way, you’ll have alternate phone numbers on hand for whenever you may need them.

Keeping the hackers at bay

The only telephone access that we provide into our system is via the IBM tie-line network (the “dial 8” network). With increased awareness and concern that was raised a few years ago about the activity of “hackers” who could possibly dial in to our system from anywhere, we discontinued the last of the outside telephone numbers.
Arrangements have been made with the telephone companies to limit the circumvention of these restrictions. For example, the IBM internal telephone system at your location may have provision for transferring phone calls, but you will probably find that you cannot transfer a data call from an outside phone into our system.

If you are ever in the position of planning activities such as an off-site meeting which is to include a terminal demonstration of your applications, be aware of this restriction of phone lines. Special arrangements would have to be made with the IBM Security department at your location to have an off-site IBM tie-line phone installed for such a meeting, or to set up special switching arrangements. Certainly, if special requirements like that are ever needed, please get us involved.
Signing On

When signing onto APLSV (or any other system), you should take precaution that your sign on number and password are not being exposed to view by others.

Signing on with a dial-up terminal

The first thing that you'll need to have to sign on is a telephone number that will connect your terminal into our system, as we discussed back on page 19. Armed with that number, you will now be able to make contact with our system. (We will not go into the details of how to use your particular terminal and modem or acoustic coupler here; that information is outside of the scope of this manual.

From any dial-up terminal, enter just a right-parenthesis [“)”] and press CARRIAGE-RETURN to hide the next input, like this:

```
)
```

(Refer also to page 34).

Then sign on normally, entering your sign-on in the form of right parenthesis, account number, colon, and your private password, like this: “)”123456:ABCDEF”.

Signing on with a 3270 display terminal

On Kingston APLSV, you can sign on from one of two places, either the “mountains” screen or the APL “apple” screen. No matter which way you choose to sign on, your sign on information can be inhibited (“blotted” out).

To sign on from the APL “apple” screen, enter just a right parenthesis [“)”] on a line by itself, and press ENTER. Whatever is typed on the next line will be hidden.
The right-paren rule also applies during your session; see page 34.

To sign on from the "mountains" screen, enter PA2 to inhibit your sign on. Type your sign on information at the bottom of the screen and depress PF1. Signing on this way will bypass the APL "apple" screen and "load-balance" you to one the APLSV systems... by "load-balance", we mean that there are monitors which continually measure the speed of operation of all of the APL systems, so that when you press PF1, you will be automatically routed to the currently fastest system.
Understanding Sign-on Error Messages

Contrary to some folks' fears, you are not faced with an infinite number of error messages that you can get at sign-on time; there are just seven (plus no message). In possibly increasing degrees of severity, they are:

- (no message at all)
- RESEND
- NO APL PORTS AVAILABLE
- INCORRECT SIGN-ON
- NUMBER NOT IN SYSTEM
- NUMBER IN USE
- PASSWORD EXPIRED
- NUMBER LOCKED OUT

No message at all

The first character of your sign-on is always a right-parenthesis. [The complete format for your sign-on is shown on the next page.] If you are using a dial-up terminal, APL looks for that first character to determine what kind of terminal you are using, so that it can translate its messages appropriately, to make them readable on that kind of terminal. If for some reason it doesn’t see that first character (possibly due to noise on the phone line, or possibly because you didn’t type it), it can’t even send you a message telling you what’s wrong, because it doesn’t yet know how to translate the message.

RESEND

As with the previous case, this message only pertains to users of dial-up terminals. You have certainly had times during conversations with friends on the phone when you had to ask them to repeat what they said, because some noise on the line obscured what they said the first time. APL occasionally has the same problems as you and me [...] it’s only human, after all. When it needs to have you re-type the entire line, it just says “RESEND” to you... a bit terse, perhaps, but it simply indicates that APL was not able to understand what you typed in. This is usually traceable to line noise or a faulty connection somewhere along the line (perhaps at your terminal or phone connection). If the problem persists, check all of the cable connections between the terminal’s phone connection and the terminal itself.

If you have the same problem on all of the phone numbers that you try, you most likely have some problem with your equipment; you will have to contact the appropriate service people at your location.

If the problem goes away when you use a different phone number, that could indicate a problem with one of our phone lines. Be sure that you report that problem to the APL Help Desk (T/L 695-1234), so that they can have the phone line checked.
**NO APL PORTS AVAILABLE**

This message only occurs for users of 3270 and 3290 display terminals who connect in to APL by means of a network connection. Telephone connections into the system are referred to as "ports", and we only have a finite number of them available for network users. They have to be shared by anyone needing access, so they are made available on a first-come, first-served basis. "NO APL PORTS AVAILABLE" is sort of the system's equivalent of a busy signal.

It is also possible to see this message if you try to sign on after the system has shut down at night (at which time there are zero ports available).

**INCORRECT SIGN-ON**

This is a common one to get. It simply means that the format of your sign-on entry was wrong. It doesn't imply anything about whether your number is still active or whether your password is right. If you see this message, you haven't gotten that far yet.

The proper format for the sign-on entry is:

```
It always starts with a right-parenthesis

(123456:ABCDEF

Next, there's always a colon

Your sign-on password

Your sign-on number
(on Kingston APLSV, it's usually the same as your employee serial number)
```

If it's not in this format, you won't get any further.

**NUMBER NOT IN SYSTEM**

This one sometimes causes confusion. When you sign on, you have to supply both a valid sign-on number (account number) and a password which is valid for that account number. If either one of those two things are wrong, you will get this message. Therefore, the message may mean that the sign-on number that you entered really isn't on the system, or (more frequently), it simply means that the password that you supplied isn't currently the right one for that number.

"So why do they do this?", you probably want to know. Why not make it simple, and just say *WRONG PASSWORD* or something like that? Well, that isn't done because doing that would give out too much information to someone who is trying to break in. (Yes,
we realize that you aren't trying to break in, you're just trying to
get your job done. But occasionally, some people are trying to
break in.)

If the number and password messages were separated, someone
could simply keep trying different numbers until he got that
mythical message of WRONG PASSWORD; then he would know that
he had gotten the number right. After that, he would just have
to keep trying passwords until he guessed the one for that
account. That's why the WRONG PASSWORD message doesn't exist.
You have to get both right.

Of course, it is possible that your account number really isn't on
the system anymore. This would only ever happen if you hadn't
used the account at all for several months. If that does happen,
don't worry; all of your workspaces and files have been put on
tape for safe-keeping. We can get all of them back up to two years
later; no work will be lost. Just call the Kingston APL Adminis-
trator on T/L 695-5238. (Refer to the chart on page 30.)

**NUMBER IN USE**

Your APL account number is already signed on at another
terminal. You should be the only one who uses your account
number. If other members of your group borrow your account on
a regular basis, perhaps they should get their own sign on
number. When you do get on, "LOAD 1 FORMS" and type
"APPLICATION" to print out a blank form for getting an additional
sign-on number for them.

If you haven't authorized others to use your account, is it possible
that you yourself may have left the account signed on at another
terminal? Be careful with this; as well as possibly getting a
security violation from your site Security people, you are
exposing your APL account to mis-use by others.

If no one else uses your account, and if you haven't left another
terminal signed on somewhere else, please call the APL Hotline
(T/L 695-1234) to see what's happening with your account. If
someone is using the account without authorization from you,
1. Let's see if we can find out who is doing this (the Hotline can
   help here).
2. Let's get them off of the system (again, the Hotline can help).
3. Be sure to change your password as soon as you sign on to
   prevent this from happening again! (See pages 28-33.)

But how do you know if anyone uses your account when you're
not around? ...What if you hadn't happened to try to sign on?
That's why the "Last-Used date" is displayed when you sign on.
(See page 27.)
**PASSWORD EXPIRED**

This one's pretty self-explanatory. You are required to change your sign-on password *at least* every two months. If you don't, the system will automatically prevent you from signing on.

This can be handled with one phone call to the APL Hotline, T/L 695-1234. They will remove the restriction for today only, so that you can sign on with your old password and change it to something else. If you don't know what your old password was, all that Administration can do is to *mail* a new password to you; passwords can never be given out over the phone. (See pages 28-33.) But assuming that you still *do* know your password, one phone call will get you going again.

**NUMBER LOCKED OUT**

Access to your account has been denied. If you get this message, the account *is* still on the system, but you are not allowed to use it. This can happen for any of several reasons:

1. If you let your sign-on password lapse and then didn't contact anyone for an additional month after the password expired, you will get this message. This indicates that your account is getting set for deletion—but it hasn’t been deleted yet, so just a phone call to APL Administration will rescue it. (Refer to the chart on page 30.)

2. You or your manager may have sent in a Change Form, requesting that the account be deleted. If so, this is the first step.

3. APL Administration may have explicitly locked this account due to unresolved problems, such as a billing dispute.

4. If you don’t respond to the APL Bi-Annual Re-Validation Audit, your account will be locked out. That’s really our only means of ensuring compliance. If the account is locked due to the Audit, it will take more than just a phone call to get it unlocked; perhaps that’s obvious. You will have to send in a completed Audit form (which Administration will be happy to send to you, via mail or VNET).

If the account is locked out, the effect to you as the intended user of account is the same as the PASSWORD EXPIRED case: you can’t sign on. However, there is one additional restriction which arises with this escalation of the withdrawal of service. On an APLSV system like this one, users can freely use each other’s workspaces (as long as they know the account number, workspace name, and workspace lock). If your password has expired, your cronies can still use your workspaces; that’s usually a transient condition. But once the account has been locked out, it is frozen: no one can use any of your workspaces. An attempt to `)LOAD` a workspace from an account which has been locked out will result in a reply of *IMPROPER LIBRARY REFERENCE*. If you have workspaces which other users in your group routinely load from your account, be sure to keep your access to the system up to date.
One-Minute Sign-On Time-Out

Any dial-up customer who does not successfully sign on to the system within one minute of the initial phone connection will be dropped automatically from that phone line. This prevents unlimited attempts to sign on under a single dial-up.

Likewise, if you haven’t completed the sign-on process on a 3270 terminal within one minute, the APL “apple” logo will also time out.

Logging of Unsuccessful Sign-On Attempts

Unsuccessful sign-on attempts are logged by the system, to enable us to trace attempted misuse of the system. The log records that are created by these attempts are reviewed each day by APL Support.

“Last-Used Date” and “Password Expiration Date”

The date of the last use of your account is displayed at sign-on time, as an aid toward better security control of the use of your account. (How else would you know if someone had been using your account last week while you were out frolicking on the beach in Altoona?)

Also displayed at sign-on time is the date that your sign-on password will expire. Please remember that your password must be changed at least once every 63 days to prevent our security monitor from automatically locking the account out of the system on the 64th day. (This is discussed in detail on the next few pages.)

A typical APL session sign-on looks like this:

```
OPR: IMPORTANT ANNOUNCEMENT: ALL USERS )LOAD 1 NEWS 074) 14.48.22 10/29/84 KJDOE - KINGSTON, SYSTEM H
LAST SIGNED ON 10/26/84, PASSWORD EXPIRES 11/29/84
IBM BUSINESS USE ONLY
APL S V IC.
SAVED 16.30.11 10/26/84
```
Controlling Your Sign-On Password

Your sign-on password should never be disclosed to another person, and ideally, never written down.

All new accounts are supplied with a randomly-generated password. We recommend that you change it once a month. You must change your password within 63 days. If it has not been changed at the end of 63 days, you will lose access to your account on the 64th day.

Your APL sign-on password may be changed by using the )PASSWORD command. Used without an argument, it returns the date by which the current password must be changed:

)PASSWORD
EXPIRES 04/10/84

To change your password, you must supply both the old password and the new password that you wish to start using:

)PASSWORD oldpassword:newpassword
EXPIRES 06/12/84

The date that is returned following that operation is the expiration date of the new password.

If the "old" password that you enter doesn’t match the one that's currently in use on your account, you will get a response of "OLD PASSWORD INCORRECT". You could also get a message of "NEW PASSWORD UNACCEPTABLE", usually due to the proposed new password being too short. In order to be in compliance with the security rules, a new password must be at least six characters long. It may be made up of any combination of A-Z, α-ω, 0-9, $, and $.

Passwords may be up to eight characters long... if a longer one is mnemonic meaningfully to you, it's usable, but only the first eight characters will actually get used by the system.

“What happens if I don’t change my password?”

If you do let your password lapse, don’t worry; assuming that you call within a reasonable time after the password expires, it can still be resolved quickly. A phone call to the APL Hotline is all that's needed... they're on T/L 695-1234. They will want to know only your sign-on number, never your password. They can then assign a grace period for signing on with the old password, so that you can get on and change it. That grace period extends only through the current day; by the next morning it will have expired again. Therefore, if you are given a grace period, be sure to sign on and change your password right away.

If you don’t call Administration within another 30 days after your password lapses, the account will be locked (you can’t sign on, and no one else can use the workspaces). If you still don’t contact Administration within 30 days after that, they will have to assume that you've gone away, and will archive the account to tape, where the workspaces and TSIO datasets will be held for two years. If your account has been deleted in this manner,
reinstatement to the system will require filling out a new application form.

Why 63 days?

Corporate requires everyone to change their sign-on passwords at least once every “two months.” After discussions with Corporate Security, they agreed to let us interpret two months as 63 days. Since 63 days is an even multiple of seven, that means that your password will always expire on the same day of the week that you originally assigned it (if you should happen to let it go that long). Therefore, if your own routine was to change it on the first Monday of alternate months, you wouldn’t find out that it had already expired on Sunday.

Realize, though, that these time periods are the maximum permissible time periods; we always recommend that your sign-on password be changed at least once a month, even though the only enforced limit is two months. If you are storing IBM Confidential Restricted data in your account, you should certainly change it more frequently than other users might change theirs.

On page 34 we describe a method of blotting out passwords, to keep others from seeing them. In that discussion, we recommend against using that method for hiding new passwords as they are being initially assigned, such as with this )PASSWORD command. If you have been using the “blot” command with this )PASSWORD command, you should refer to that discussion.

...An alternative to sign-on passwords?

“A machine’s only a machine—isn’t it?”

“This one isn’t,” Auberson said. “This one’s human.”

...Auberson thumbed a console to life.

“HARLIE,” he typed, but before he could identify himself, the machine spat back, “YES, BOSS?” …Auberson was startled.

HOW DID YOU KNOW IT WAS ME?
I RECOGNIZED YOUR TOUCH ON THE KEYBOARD.

Auberson jerked his hands back as if stung. He stared at the typer. It was a standard IBM input/output unit. Could HARLIE really sense the difference between one typist and another on its electronic keyboard? Apparently he could. It must be the minute differences in each person’s typing.

...The silvery sphere of the typing element clattered across the page...."

—from “When HARLIE Was One”
A novel by David Gerrold, © 1972
Used by permission of Doubleday, Inc.
Password Change Requirements on Kingston APLSV

)PASSWORD old:new

Memo is mailed to you, the account owner, explaining the intent to delete your account
Memo is mailed to the users of your TSIO files, explaining the intent to delete your files

Recommended next password change
(63 days maximum)

"Eligible" * for

INHIBIT

LOCK

DELETION

time

Last Password Change
1 month
2 months
3 months
4 months

After these points, a sign-on attempt will result in:

PASSWD EXPIRED
NUMBER LOCKED OUT
NUMBER NOT IN SYSTEM

You can sign on normally

Workspaces can be loaded from another account

Your account is still on the system up to here, so you can usually be re-instated with just one phone call to APL Administration

* The actual deletion takes place after the next quarterly backup tapes are made, to ensure that a suitable two-year retention copy of all of your workspaces and TSIO datasets will be available.
Proper Use of Passwords

All APL users should be aware that a password on their sign-on only protects the sign-on from unauthorized use and only protects workspaces associated with the sign-on from unauthorized modification. The sign-on password does not protect against unauthorized access to workspaces associated with the sign-on. Protection of sensitive data stored in an APL workspace is possible only by associating a password with the workspace.

Your sign-on must be protected by a password (in fact, there is no way to remove it and run with no password). Any sensitive workspaces should also be protected with passwords, and those passwords then take on the classification of the data that they are protecting (...if you are storing IBM Confidential Restricted data, your password is also IBM Confidential Restricted). We further recommend that the password that you use for signing on to APL be different from the password associated with your workspace.

We caution against using passwords such as initials, names of relatives or significant personal dates. Such personal information is more readily available than one might expect, and is specifically prohibited by Corporate Security Document 104A.

That document requires that passwords be “randomly selected, and neither obvious, nor trivial, nor predictable.” Ideally, the password should be something that you can easily remember (so that you won’t have to keep referring to a slip of paper... which perhaps someone else could get to). But it shouldn’t be anything that someone else who knows you would guess. Using a randomly-selected, purposely-misspelled word is okay, but do not use terms that are identifiable with you (such as the make of your car or your dog’s name). Believe it or not, those can often be guessed.

The use of the name of the month or your initials or nickname are common and easily guessed by anyone who wants to compromise the system. More esoteric, but still easily determined, are your spouse’s birthday, your anniversary, your child’s or pet’s name, or even the name of an old acquaintance. We recommend against any of these approaches.

Good passwords can also be constructed from memorable phrases. This is an excellent way to get rid of that tune that bothers you night and noon. For example, "GGTLASWD" for “Green Grow the Lilacs All Sparkling With Dew." ...Impossible to break (unless you whistle while you work).

Even better is the inclusion of one or more numeric digits with any of these schemes. But—as with characters— the numbers shouldn’t have any particular significance that anyone else could guess. Any numbers that you include shouldn’t represent the month and shouldn’t simply be sequentially-incremented from last month’s password.

---

4 Shown for comparison only. Your taste in music may differ, and will probably be higher (except in California).
If you don’t choose to be so original, workspace 10 SECURITY can be used to generate random character strings (shown on the next page). Remember, however, to make a protected record of your password. If you forget it, the APL Administrator will have to assign a new password and have it mailed to your manager, leaving you without APL service for a few days. Be aware also that the APL Administrator can assign a new password to your account, but not to a workspace.

•

"Why should I have to lock my account? There’s nothing in it!"

Any user on the system can load workspaces from another user. If someone signs on to your account, don’t think of the exposure simply in terms of your own material... he can access other people’s material. Unauthorized access, possibly even by an outside (non-IBM) user, could jeopardize the security of major applications.

“So what? I don’t care!”

Then let’s get a little bit more practical about this, and see if we can make it relate. Even aside from the security aspects involved if someone else gets hold of your password, any billing that they accrue will be sent to you. You wouldn’t lend out your credit card, would you? The same consideration, of course, is involved in lending your account to one of your co-workers: we don’t recommend it. If your department needs more accounts, "LOAD 1 FORMS" and type "APPLICATION" to print application forms. If someone else uses your account, they could use the system without being accountable for its use (...you would be accountable).

•

The Bottom Line

The protection of any system ultimately comes down to a sign-on password. No matter what else in this manual you pursue, no matter what other steps you take, your applications can only ever be as secure as the protection that you provide for your sign-on password.

If you are in charge of some “super-secure” application, but then let others see or guess your sign-on password, the system will of course assume that they are you if they should sign on with your account number.

Furthermore, even protecting your sign-on password is for naught if you then walk away and leave your terminal signed on and unattended. Again, passers-by could do anything with your account that you can.

Exercise care when you choose a sign-on password, protect it well, and don’t ever leave an active terminal unattended. ...It’s really the bottom line for everything else that’s discussed in this manual.

6 If you would, please let us know... our car payment is due again.
Generating Random Passwords

We feel that it's best to choose a password that's meaningful to you, so that it can be easily remembered without writing it down (...a randomly-selected password). However, if you are intent on generating truly random passwords, workspace "10 SECURITY" has a function that will do it for you:

```
PASSWORD∥v
v 2+PASSWORD;R;N;ALF;[]I0;[]RL
[1] CREATE RANDOM PASSWORDS (FOR APL WORKSPACES OR SIGN-ONS)
[2] []I0+1
[3] LIMITED ALPHABET RELIEVES CONFUSION BETWEEN '0' AND 'O'. ETC.
[4] ALF+"ABCDEFGHIJKLMNOPQRSTUVWXYZ2346789"
[5] []RL++/[]TS
[6] []RL??/[]AI
[7] ALF+ALF[[]R;[]R+pALF]
[8] 2+ALF[6;R]
[9] IF YOU WISH TO HAVE RANDOM-LENGTH PASSWORDS, USE THIS:
[10] N←6 7 8[[]3]
[11] 2+ALF[N;R]
```

Let's run the function a few times to try it out:

```
PASSWORD
WYTKD4
PASSWORD
KD9ZNG
PASSWORD
JXT23R
6 6pPASSWORD, PASSWORD, PASSWORD, PASSWORD, PASSWORD, PASSWORD
MGT4DR
HN8AEK
AG23RE
PKM2WZ
FECX4G
GB76UC
```

Sign-on passwords must be protected with at least the same measure of protection that is required for the data that they are protecting, with "IBM Confidential" being the minimum classification (since the system itself must also be protected).
“Display Inhibit” for Hiding Sensitive Inputs

Entry of a right-parenthesis only, at any time, will cause the next input to be hidden from view. Its uses include:

- Hiding your password at sign-on time
- Hiding a workspace password during )LOAD or )COPY operations
- Hiding any sensitive data

...or perhaps you just don’t want that chap standing beside you to see which workspace you are loading.

On a 3270 Display Terminal, entry of a right-paren-only will put the terminal into “print inhibit” mode, so that the following line does not display as you type. For example:

```
) entry of a right-paren...
...hides the next line.
SAVED 16.20.15 12/08/84
```

See below for a discussion of Display Inhibit in full-screen mode.

Beware— This is NOT designed for entering NEW passwords!

We should point out that we do not recommend using this command for hiding a new password when you’re entering it with the )PASSWORD command, because the chance for mis-typing it is too great. If you enter other than what you think you’re entering, you will not be able to sign on the next time, and we will have to mail you a new password (...we can’t ever give out passwords over the phone). For entering a new sign-on password (or for assigning a new password to a workspace), you’re better off to just close yourself up in your office or terminal room, where no one else can see your screen or page, and enter your new password carefully.

Also remember to discard printed output in Confidential bins, and if you are using a 3270, sign off after entering a new sign-on password to clear the session logs (so that someone can’t page back and look at the password if you step out of the office).

Dial-Up Terminal Users Only

On all hard-copy dial-up terminals, a “blot”-pattern of over-struck characters is printed to obscure the next input:

```
)
)LOAD
SAVED 16.20.15 12/08/84
```

Please be aware of a security exposure that will always exist: if you are one of the vanishingly few people who have a typewriter terminal (such as perhaps a mag-card terminal) with a carbon ribbon, everything that you type is readable from the ribbon. Use of the blot will not change that; it’s a hardware limitation that APL cannot change. Therefore, if you are typing something which is truly of a sensitive nature, you would be well advised to open the terminal cover, and put the ribbon in the “stencil” position before entering it, or if it is more than just a single entry, treat the ribbon just the same as you would treat any other IBM
Confidential material. Please note that this concern is related only to carbon ribbons, not to fabric ribbons or to the Tech-III ribbon.

**Display Inhibit in Full-Screen Mode on a 3270**

The “format” statement that’s used with AP124 (the full-screen auxiliary processor) to format the various fields on the screen can be set up to include a non-display field, so that data which is entered from the keyboard gets passed to your program but will not display on the screen as it is being entered. This duplicates the facility which is used at sign-on time on a 3270, to hide your sign-on password.

The data variable may be “n” rows by 4, 5, or 6 columns. If `DAT` is a vector, it is treated as a 1-row matrix.

**Screen Intensity:**
- 0—off or don’t display
- 1—normal intensity
- 2—highlighted

**Field type:**
- 0—character input/output allowed
- 1—numeric input/output allowed
- 2—character output only
- *—character output/light pen interruptable
- 3—character output/light pen selectable

Complete details were published in the Fall 1980 issue of The APL Jot·Dot Times [the “Special Reference Issue”].
Here is an example of a function which prompts the user for input using full-screen mode to accept entries in a non-display field:

```
Z INT BLOT MSG R A CTLS DATS H W ROW COL L M V IO

DISPLAYS (VECTOR OR MATRIX) MSG IN RT ARG; TAKES IN HIDDEN INPUT

LEFT ARG: INTENSITY OF PROMPTS (0, 1, OR 2) -- OPTINAL

IF NOT SUPPLIED, IT DEFAULTS TO: 2

MAY BE ONE OR TWO ELEMENTS (FOR TWO FIELDS)

RIGHT ARG: TEXT FOR PROMPTS; MAY BE A VECTOR OR A MATRIX

IO+1

MSG+(-2+1;pMSG)pMSG ← Turn input messages into a matrix

ESTABLISH NEW SCREEN FORMAT

'NO SHARE' 'A+12'B SOVO 2 p 'CTLSDATS' ← Offer shares

A+1 SVC 'CTLSDATS' ← Request tight interlocking of shared-variables

H+1;pMSG ← Height of messages

W+1;pMSG ← Width of messages

H INT 1 'INT' ← Default left argument for monadic call

VALIDITY-CHECK LEFT ARGUMENT

VAL INT 'INT' ← Scalar extension

ROW 0.5x24-H ← Calculate centering of the message

COL 0.5x80-W

L 81-COL

V (+((ROW+1),COL,(H-1),W,2,1+INT),(ROW+H),COL,1,L,0 0

DATS+M 3 6 p,(ROW,COL,1,W,2,1+INT),V ← Create formatting matrix

FORMAT THE SCREEN

'0' (0*1Z+CTLS)/0 ← If other than 0 return code, exit

SET CURSOR POSITION: NUMBER OF SCREEN FIELD, FIELDNUM, COLUMN

DATS 3 1 1

CTLS 12

(0*1Z+CTLS)/0

SET FIELD 1

DATS MSG[1;]

CTLS 4 1

(0*1Z+CTLS)/0

SET FIELD 2

DATS 1 0 +MSG

CTLS 4 2

(0*1Z+CTLS)/0

SET FIELD 3

DATS Lp ' '

CTLS 4 3

(0*1Z+CTLS)/0

READ DATA ON SCREEN, AND RETURN STATUS VECTOR

CTLS 3

(0*1Z+CTLS)/0 ← If other than 0 return code, exit

A+DATS

GET DATA IN FIELDS IN FIELDNUM

CTLS 5 3

(0*1Z+CTLS)/0

Z+ DATS

A+SVR 2 4 p 'CTLSDATS'

Z+(-/-/A\$Z=' ')'+Z ← Remove extra blanks from user's input

IF SVO ENCOUNTERED INTERFACE QUOTA EXHAUSTED, Halt

36

Section 3: Protecting Your System Sign-On
```
Section 4: Protecting Your APL Workspaces

Workspaces and Libraries

The common organizational unit in an APL system is the workspace. When in use, a workspace is said to be active, and it occupies a block of working storage in the central computer. Part of each workspace is set aside to serve the internal workings of the system, and the remainder is used, as required, for storing items of information and for containing transient information generated in the course of a computation.

A terminal always has an active workspace associated with it during a work session, and all transactions with the system are mediated by it. In particular, the names of variables (data items) and defined functions (programs) used in calculations always refer to objects known by those names in the active workspace; information on the progress of program execution is maintained in the state indicator of the active workspace; and control information affecting the form of output is held within the active workspace.

Inactive workspaces are stored in libraries, where they are identified by arbitrary names. They occupy space in secondary storage facilities of the central computer and cannot be worked with directly. When required, copies of stored workspaces can be made active [by using the "\LOAD" command], or selected information may be copied from them into an active workspace [by using the "\COPY" command].
The Effects of Selected System Commands and TSIO

Your Terminal

(Encryption is available for terminal lines)

Your Active Workspace

Instructions or data

)CLEAR

)LOAD

)SAVE

(Moves the entire workspace and replaces everything previously stored there)

)COPY or )PCOPY

(Additive—adds all or selected objects from an inactive workspace to the contents of the active workspace)

Libraries of Stored Workspaces

Special pre-initialized (empty) workspace: "CLEAR WS"

TSIO data

SR, IR, or IRW

SW or IRW

TSIO moves data between the active workspace and a dataset.

Datasets (not workspaces) containing essentially unlimited storage capacity.

All of the access to TSIO datasets, whether through APL using TSIO or through batch, is mediated by RACF. This allows control of not only who can access the data, but also how each of those authorized users may access it.

Other languages on other systems (MVS, TSO, CMS, and so forth) can share the use of these same datasets.

Other languages on other systems (MVS, TSO, CMS, and so forth) can share the use of these same datasets.
Libraries of Saved Workspaces

The set of workspaces that you have saved is called your library. Each workspace is identified by your account number and the name that you assign to it. However, in referring to workspaces in your own library, the account identification may be omitted; your own identification is then supplied automatically.

Rather than always re-inventing the wheel, it is often convenient to use functions or variables contributed by others. You can activate an entire workspace saved by someone else, or may copy selected items from it. To do so, both the library number and the name of the desired workspace must be supplied (along with the password, if there is one... more on that in just a moment). However, APL provides no way of learning either the account identification or the names of workspaces belonging to other users. Thus, you may make use of material from the libraries of others only if they supply that information.

In no case may you add, change, or delete material from someone else's library.

Libraries 1 through 999 are not assigned to individual users, but are designated as Public Libraries. Any user may obtain a list of workspaces in a Public Library, and may use public workspaces. However, only the owner of a workspace can save, drop, or modify that workspace.

The contents of any Public Library may be displayed by entering \LIB followed by the library number [as in "$LIB 1"]. Once you get its name from the \LIB-command, a description of how to use any particular workspace in the Public Library may be displayed by \LOADing that workspace and then typing "DESCRIBE". By convention, any workspace in the Public Library should have some sort of descriptive text under the name DESCRIBE.

A list of the major offerings in the public library is available: \LOAD 1 CATALOG.

Locking Your Workspaces

Stored workspaces and the information they hold can be protected against unauthorized use by associating a lock, comprised of a colon and a password of the owner's choice, with the name of the workspace, when the workspace is stored. In order to activate a locked workspace using APL or copy any information it contains, a colon and the password must again be used, as a key, in conjunction with the workspace name.

If you have some information in a workspace that is classified as IBM Confidential or higher, or if you just don't want the average user accessing your workspace, you should put a lock on your workspace. In order for anyone to be able to load your workspace, they would need to know the workspace name and its lock.
A workspace is locked with the )WSID command or with the )SAVE command. To lock your workspace, type:

```
)WSID workspacename:lock
Was workspacename
```
or:

```
)SAVE workspacename:lock
17.11.34 05/18/84
```

You must remember the lock you put on your workspaces. In this case, if you forget it, we cannot help you. The system allows the password to be from one to eight characters long, but we recommend that it be at least six characters long. This lock, as with your sign-on password, should be non-trivial.

To load a locked workspace, type:

```
)LOAD workspacename:lock
SAVED 17.11.16 05/18/84
```

If you try to load a locked workspace without specifying the lock, the system will return "WS LOCKED".

Listings of workspace names, including those in public libraries, never display the keys, and do not overtly indicate the existence of a lock.

The user who has augmented a workspace name with a password has taken the first primitive step toward application security. This type of security should be considered to be more of an ensurer of privacy than a technique to control the sharing of a workspace with others. Disclosure of the password to another user allows that user full access and the ability to make his own copy of the workspace or to disclose the password to any other users.

The primary use of a locked workspace for security should therefore be as a repository of unlocked copies of functions which are locked in other workspaces. When used in this fashion, the workspace password (like the sign-on password) should never be disclosed to another user.

**What If the System Creates a CONTINUE Workspace?**

If your active session is interrupted due to line disconnections or terminal power disruptions, your active workspace will be saved under the name "CONTINUE" and will automatically be re-loaded for you when you next sign on. If there was a lock associated with the workspace that you had been using, that lock will also be used on the CONTINUE workspace, and in that case, the workspace will not be loaded the next time that you sign on. Therefore, if you had been using a workspace called "PLAN" with a lock of "4TODAY", you would normally activate it by entering ":LOAD PLAN:4TODAY". If your session became interrupted without you properly signing off, the next time that you signed on you would need to enter ":LOAD CONTINUE:4TODAY" to recover your work.
Extra Protection for the CONTINUE Workspace

APL never allows another user to inquire as to the workspace names in another user’s account (that is, you cannot “LIB” another user’s private library). This is done for obvious reasons of privacy. Since most users often have a CONTINUE workspace, though, there has to be some extra protection to preventing a user from stealing material from another user’s CONTINUE workspace. The burden shouldn’t be on the owner to lock it, since it’s one workspace name that you don’t usually have to guess.

This is prevented by allowing any workspace named “CONTINUE” to be loaded only by the owner’s account; you cannot ever LOAD CONTINUE from another user’s account.

As a logical extension of this rule, a workspace named CONTINUE cannot appear in a Public Library.

Years ago, it was far too easy to sneak a peek at someone else’s work, if you were just a bit devious. Let’s assume that your manager was going over the salary plans on APL, and you really wanted to see what everyone earned. No problem; you’d just go out to the aisle, turn off the circuit breaker that controlled the outlet that his terminal was plugged into, and snicker as he got disconnected from APL. Then—knowing that his workspace was now in CONTINUE—you’d simply go back to your office and load it from his account. If he hadn’t yet assigned a password to the workspace, you would get it, without even having to know the names of any of his regular workspaces. Obviously, that’s not acceptable. APL hasn’t permitted that sort of fraudulent bamboozlement since APLSV was introduced in the early 1970’s. An attempt to “LOAD CONTINUE” from any account other than your own will now result in IMPROPER LIBRARY REFERENCE.
Restricted Libraries (Load-Only Workspaces)

At your request, APL Support can "restrict" a library for you. A "restricted library" contains workspaces that can be loaded but not copied, either in whole or in part. It is also not possible to save these workspaces into another account [e.g., if the restricted library is a public library, when you load a workspace from it you cannot save it into your own account]. Likewise, an attempt to rename the workspace, via \WSID, will be rejected, as will \CONTINUE, since this implies a save. Finally, if the telephone connection is broken while a workspace from a restricted library is being used, or if the user is "bounced" off of the system, the workspace will never be saved in \CONTINUE.

By restricting the workspace to load-only operation, a forced execution of any desired latent expression (\LX) is ensured. But please understand the limitations of the latent expression. It is discussed in detail on page 48.

Any attempt to bypass the security of this provision, such as entry of a \SAVE, rename via \WSID, or \COPY, will elicit an "IMPROPER LIBRARY REFERENCE" or "COMMAND DISALLOWED" message.

This restriction is not imposed upon the person who originally saved the workspace into that library; that is, he will be able to rename, resave, copy, without restriction.

It should also be emphasized that this restriction must be applied on a complete library basis... it cannot be applied to individual workspaces.

This facility provides an extra degree of "blanket coverage" protection for sensitive programs. It is a guarantee that even the users who are authorized to use the programs cannot save a copy into their own libraries. Why would you care about this? ...Well, if you discover a particularly devastating bug in your code someday, there's at least a modicum of peace in knowing that once you fix your single library copy, you have solved the problem; there can't be any other down-level copies floating around in various users private libraries, lurking around, waiting to destroy your database again tomorrow. While this does not in itself make an application secure, it helps.

An example of a restricted library is Library 1; while we encourage everyone to \LOAD \NEWS every day, you cannot \COPY items from \NEWS. Library 1 has some sensitive material in it; this adds that little extra measure of protection to it.

If you would like additional information on this, or if you wish to restrict a particular library for which you are responsible, call the Kingston APL Hotline, T/L 695-1234, and ask for some programming assistance.

---

6 Also sometimes called a "secured library".
Confined Accounts (for Non-IBM Access)

Kingston APL has some accounts on the system which do not belong to IBM employees. These include consultants for projects within the company, vendors keeping inventories of the parts that they are supplying for IBM, universities doing studies for IBM, and even some accounts that are further removed from daily business, such as Boy Scout training programs. Obviously, we have to have mechanisms in place to be able to guarantee that these accounts cannot be accessing IBM material that wasn’t set up for their use.

An APL account which has this control set up for it is referred to as a “confined” account. A confined account may only ever access the workspaces in its own private library, or the workspaces in selected public libraries.

Notice that this means that they can’t even access each other’s workspaces. On future services that we offer (such as APL2), this protection will be provided by RACF, which will finally give us the flexibility to allow separate groups of users to access each other’s workspaces within their own group, but can prevent them from accessing other groups.

If the user of a confined account tries to )LOAD a workspace from another private user or from a public library which we have not designated for his use, he will receive a message of “IMPROPER LIBRARY REFERENCE”. The same response will be given if the user issues a “)LIB” command to list the names of the workspaces in other than the designated public libraries. He is not only prevented from loading the workspace, he is also prevented from even being told of the existence of those workspaces.

In this way, the user is “confined” to his own particular area of operation, and the rest of the system just doesn’t appear to exist to him.

IBM Open House or Family Day Arrangements

If your IBM site is planning to hold an open house or family day, and you have been assigned the task of setting up terminal demonstrations on Kingston APL, get us involved. We strongly recommend against simply signing on one of your standard accounts and letting the world use that. Attendants will always get called away at times, and all it takes is one mishap by one visitor to sour management on the idea of offering system access again. Those problems are easily preventable; contact APL programming support, and have us set up a new confined account for you. It removes the worry.
Section 5: Protecting Your APL Functions


Section 5: Protecting Your APL Functions

Using System Variables

It is certainly not possible to describe all of the measures that can be taken to ensure that the person who executes your code is authorized to do so, because the requirements of what to check differ by application and by imagination. You have the full power of the APL language available to you for disguising data, checking user authorizations, and so forth. The APL system variables provide a wealth of information about the user, his environment, and his current session. Shown here are just some examples of common types of things to check; you may well have additional needs or ideas.

Localizing System Variables

If your application is depending upon the value of some of the user-settable system variables (such as CT, IO, and FF), it is important to localize these variables in your functions. If you don’t localize them, it would be possible for a knowledgeable user to change their value and perhaps affect your calculations... possibly even to the extent of forcing your code to return invalid data.

Even if your code doesn’t use the system variables directly, you should decide whether or not your code would be affected if they were changed. For example, you may not set the Index Origin (IO) in your own code; you may simply depend upon it being set to the system default value of 1. But what happens if a user of your application changes it? Would that affect your code? ...Possibly it would. Be especially careful of some branch statements that may suddenly branch in the other direction if the index origin is changed. If this were to occur in your own code, you need to decide if it is possible that your personnel salary reporting package could show a user the salary history of everyone except himself! And would that be a problem if it did? A way to get around that is to localize it and set it to the value that you’re expecting.

Another variable to be cautious of is the Comparison Tolerance (CT). There are always some numbers that cannot be exactly represented on a digital machine (...in fact, there are more than those which can be precisely represented). To facilitate comparisons, APL lets you tell it that close is “close enough” when you’re comparing two numbers. If they are the same out to
thirteen decimal places, you may want to consider them to be equal (even though, in fact, they may not be exactly equal). It’s a very helpful facility in APL—when it’s used properly. Comparison to thirteen decimal places is the default value, but you can specify your own tolerances. If some user specifies an overly-tolerant tolerance, you may find that suddenly everything is equal. This could be quite a sticky wicket for applications which compare a single value against a list of values, to return matching data. For example, in order to determine who you are and which mail is yours, the APL Mailbox has to encode your mailcode into a numeric value, and compare it against a list of all of the users. Suppose that a bizarre value of OCT caused that comparison to match every entry. Delivering everyone’s mail to you would be decidedly unpopular—with the senders, with the recipients, and with Security. Be sure that this value is localized and set to a meaningful value in sensitive code.

In applications in which you may format and execute numbers—not generally a good procedure, by the way—you should also consider localizing the Printing Precision variable (OPP).

By the way, contrary to what has occasionally been published in some other literature, APLSV does not provide a facility for reading the terminal-ID from within your application. We have now seen that published in a couple of other user’s guides (always without the details of how to actually do it). Unfortunately, APLSV never has had that facility.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description and Typical Uses</th>
</tr>
</thead>
</table>
| **AI** | **Accounting Information:**  
   1. User number (APL sign-on number)  
   2. Compute time this session (in milliseconds)  
   3. Connect time this session (in milliseconds)  
   4. Keying time this session (in milliseconds)  
   Obviously, the most common use for this with respect to security is to check the user number against a list of authorized users. If you do this, the check should be in-line code in the function that you want to protect (not just in the latent expression). See page 48. |
| **LC** | **Line Counter:** statement numbers of functions in execution or halted, with the most-recently-activated shown first. Used sometimes to determine whether or not it appears likely that a given function was called from within a standard calling sequence, or if it was just called individually. That procedure is not very reliable, since a user could simply call it from within a different function. |
| **LX** | **Latent Expression:** executed automatically when the workspace is loaded. This provides a convenient facility for providing instructions to the users of an application, or to let the application “come up running” when the workspace is loaded, but it is often grossly mis-used as a security measure. Refer to the discussion entitled “Use and Mis-use of the Latent Expression,” on the next page. |
| **RL** | **Random Link:** used for the generation of pseudo-random numbers by the APL “roll” (\( ?n \)) and “deal” (\( n?n \)) functions. See our discussion of this on page 109. |
| **TS** | **Time Stamp:**  
   1. Year  
   2. Month  
   3. Day of the month  
   4. Hour (on a 24-hour clock)  
   5. Minute  
   6. Second  
   7. Millisecond  
   This can be useful for applications in which you need to ensure that the code cannot be run during certain times, such as off-shift or on weekends. It does not take into account differences in time zones; Kingston APL is a world-wide service, but the time stamp always shows Kingston local time. Also, if your intent is to allow operation only during IBM working hours, you will also be responsible to set up your own calendars for taking IBM holidays into account (which also differ by IBM location). |

Further information is available regarding these system variables and others; see The APL Language Manual (GC26-3847).
Use and Mis-use of the Latent Expression (□LX)

The APL statement represented by the Latent Expression (□LX) is automatically executed whenever the workspace is activated with a "")LOAD" command. Formally, □LX is used as an argument to the execute function (♀□LX), and any error message will be appropriate to the use of that function.

Here are some common forms of the latent expression:

To invoke an arbitrary function F:
   □LX+' F'

To print a message upon activation of the workspace:
   □LX+' 'FOR NEW FEATURES IN THIS WS ENTER: NEW''

To automatically restart a suspended function:
   □LX+' -+□LC'

The variable □LX may also be localized within a function and respecified therein to furnish a different latent expression when the function is suspended. For example:

□LX+'START'

♀START;□LX
[1] □LX+'-+□LXL-flf WE CONTINUE FROM WHERE WE LEFT OFF''
[2] 'WE NOW BEGIN LESSON 2'
[3] Drill□FUNCTION
[4]♀

)SAVE ABC
19.24.32 11/29/84

On the first activation of workspace ABC, the function START would be automatically invoked; if it were later saved with START halted, subsequent activation of the workspace would automatically continue execution from the point of interruption.

Additional information is available in The APL Language Manual (GC26-3847).

It sounds easy so far, but BEWARE....

Some users have gotten themselves into trouble by assuming that this facility could be used for security checks as the workspace is loaded. We sometimes see this somewhat dangerous construction:
...where CHECK looks like this:

```
v CHECK;A
[1] A...AN EXAMPLE OF A BAD APPROACH; DON'T DO THIS.
[2] A
[3] A LIST OF AUTHORIZED USERS:
[4] +(~1+D)D'123456 234567 345678 456789)/NO
[5] OKAY::'WELCOME TO THIS FACILITY; TYPE: START'
[6] +O
[7] NO::'UNAUTHORIZED ACCESS'
[8] A EXPUNGE ALL OF THE FUNCTIONS AND VARIABLES:
```

*This approach is not recommended!* It suffers from several serious problems:

**Problem Number One:**
The latent expression is *only* invoked if you `LOAD` the workspace—*not* if you `COPY` the workspace. No, that's not a bug; that's the way it's supposed to be. (It is possible, though, to have the library set up so that the workspaces can only be loaded and not copied. Refer to the discussion of Restricted Libraries on page 42.)

**Problem Number Two:**
Line 7 of the function expunges [dynamically erases] all of the functions and variables in the workspace. That's being done in an effort to simulate the action of `CLEAR` [which can't be executed from within a function]. But realize that that's not quite the same. In particular, the workspace name is not changed to "CLEAR WS". That may sound trivial, but if you are the programmer who owns this workspace, you run the risk of forgetting to include your own account number, running the function, expunging everything, and re-saving the workspace—discarding all of your work in the process.

**Problem Number Three:**
Since the expunge function (EX) can't currently expunge the function that it is invoked from, the CHECK function is left behind. If it is not locked, a user could display the function and get a list of the authorized users. While that may not be directly harmful, you should generally limit distribution of authorized account numbers to those people who have a need to know.

And, as if those problems weren't enough....

**Problem Number Four:**
[The *coup de grâce*]: No one ever said that the latent expression couldn't be interrupted, and in fact, it can be—quite easily. If the user who loads this workspace is quick with the ATTENTION key (or PA2 key) he can halt the operation of the CHECK function before it does its checking. Remember, the point of the CHECK function was supposed to be to keep...
random users from seeing what's in the workspace. *This won't do it.*

Picture the following scenario: You are lying in bed, late at night, and you hear someone prowling around through your house. So you go downstairs, see if he's supposed to be there, and if not, put him out and lock the door. Does that sound like a reasonable approach? **NO!** By that time, it's *too late!* If you really wanted to keep him out, you should have locked the door. And yet, this is exactly the same logic that you would be putting into your code by keeping a list of authorized users in the workspace, and then attempting to discard everything if the user isn't authorized to see it.

*A much better approach* would be to use a command dataset (described on pages 85-91) to selectively allow authorized users to open a dataset, and *if* they are authorized, they can access your sensitive data. If they aren’t authorized, there shouldn’t be anything in the workspace that you would be concerned about them seeing.
Erasing Functions and Variables Dynamically (EX)

The expunge function, EX, provides the facility to dynamically eliminate an existing use of a name. By “existing use”, we mean the current, most local copy. Thus, EX 'PQR' will erase the most local copy of the object PQR unless it is a label or a group; those may not be expunged. As one example of its use, certain name conflicts can be avoided by using this function.

The function returns an explicit result of 1 if the name is now unencumbered, and a result of 0 if it is not, or if the argument does not represent a well-formed name. A result of 1, therefore, signifies that the name is available for use, whereas a 0 signifies that it may not be used.

Expunging a shared variable will retract the sharing and erase the variable.

The expunge function applies to a matrix of names and then produces a logical vector result. EX will report a RANK ERROR if its argument is of higher rank than a matrix, or a DOMAIN ERROR if the argument is not a character array.

You may wish to compare the operation of EX with that of the "ERASE" system command, (discussed in The APL Language Manual (GC26-3847)). One major difference (other than form) is that EX discards the most local reference to a name that is currently active (which may be a global object), and ERASE discards only global references.
Locked Functions

At times, you may want to allow other users to access a workspace while keeping them from disseminating the particular procedures or algorithms that you used in solving the problem. Seeing the answer that you arrived at in solving a problem may be of interest to them; seeing how you arrived at that answer may be of more interest, and on occasion, may be something that you want to conceal (perhaps pending publication).

For those cases where you have discovered *The Meaning of Life*—but don’t necessarily want everyone to see how you have determined it (even if they already happen to know the answer), APLSV provides a function locking facility. A locked function may be executed just like any other function, but it cannot be displayed or suspended, and thus cannot be examined by *any* user (including you).

The function locking facility differs from the workspace locking facility in two ways. Locking a workspace prohibits access by any unauthorized person, but does not prevent authorized users from examining the contents of the workspace. No authorization is inherently required for executing a locked function, but no individual, *including the originator*, can unlock the function to examine it or to change it. Once locked, it remains locked until erased. It is important to maintain an unlocked copy of the function in a separate workspace in order to be able to modify, update, or examine the process. Small amounts of critical data can be protected by including their localized specification within the locked function in which it is used.

The procedure for locking a function is simple. If the symbol “*" (called del-tilde or protection del) is used instead of “v” (called del) to open or close a function definition, the function becomes “locked”.

A locked function cannot be revised or displayed in any way. An attempt to display the function will result in a “definition error”:

```
vMOL[] v     ← displaying the function
  v MOL
[1] THE MEANING OF LIFE
[2] DO NOT RELEASE PRIOR TO ONTOLOGICAL SYMPOSIUM
[3] Z+0[+1×!aPp],b-c0
  v
vMOL* v     ← locking the function
vMOL[]v v   ← trying to re-display it
DEFN ERROR
vMOL
  v
MOL        ← ...but of course, it can still be used.
```

Section 5: Protecting Your APL Functions
Any associated stop control or trace control is also nullified after the function is locked.

Once a function is locked, it cannot be unlocked again; locking is permanent. If you lock a function, be sure that you keep an unlocked copy of the function in another workspace. A common method of handling this is to keep the unlocked functions in a locked workspace, and to give other people access to the locked functions in a separate unlocked workspace:

<table>
<thead>
<tr>
<th>WSID:</th>
<th>MEANINGUNLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK:</td>
<td>BANANAS</td>
</tr>
<tr>
<td>FNS:</td>
<td>&lt;MOL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WSID:</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK:</td>
<td>(none)</td>
</tr>
<tr>
<td>FNS:</td>
<td>*MOL</td>
</tr>
</tbody>
</table>

Your users should not even be aware of the existence of this workspace... but even if they discover it, they can't LOAD it.

Locking a function does these four things:

1. Prevent the display of the function (use of the built-in editors will return DEFN ERROR, and OCR will return a 0 by 0 matrix).
2. Prevent suspensions, as is done with primitive functions.
3. Ignore weak interrupts (attention) during its execution.
4. Convert error messages to DOMAIN ERROR, to further hide its internal workings. [Resource and environment errors, such as WS FULL, will continue to be reported.]

A locked function is treated essentially as a primitive, and its inner workings are concealed as much as possible. Execution of a locked function is terminated by any error occurring within it, or by a strong interrupt. If execution stops the function is never suspended but is immediately abandoned. The message displayed for a stop is DOMAIN ERROR if an error of any kind occurred, WS FULL and the like if the stop resulted from a system limitation, or INTERRUPT.

**Tying Your Shoes**

You've probably all observed this human phenomenon: if you are interrupted while you are tying your shoes (an obviously primitive operation), there's just no way that you can resume from where you left off: you will undoubtedly have to untie them and start over again. Well, locked functions are just the same as you and me. [Tying your shoes was the model for locked functions.] If a locked function is interrupted during the course of its execution, execution is abandoned, and the function is not suspended. A locked function which has been interrupted will have to be restarted from the beginning—consider this if you are
planning to lock the functions which are used for a long-running process.

When a locked function is interrupted, the error line will be where the function was called. If a locked function was invoked by an unlocked function, that unlocked function will be suspended; if it was invoked by a keyboard entry the error message will be displayed with a copy of that statement.

Similarly, when a weak interrupt is encountered in a locked function, execution continues normally up to the first interruptable point: either the next statement in an unlocked function, or the completion of the keyboard entry that used this locked function. In the latter case, the weak interrupt has no net effect. When a weak interrupt is encountered, any printing which would otherwise have occurred at the terminal is discarded, but execution of the function continues, either until the execution sequence is complete or until a second interrupt is received.

Locked functions may be used to keep a function definition proprietary, or as part of a security scheme for protecting other proprietary information. They are also used to force the kind of behavior just described, which sometimes simplifies the use of applications.

**Throw Away Your Old Building Blocks**

It really pains us to have to tell you this, but if you wish a locked function to be secure, it must not call any other functions or reference any global variables. *Heresy!* ... A few years ago, we devoted an entire issue of Jot to a discussion of recommended programming styles, and the cover story was, “How to Use Building Blocks.” That was our “Access to Tools” issue, and we pointed out that a building-block approach “can result in some rather substantial benefits to you”. Well, we still feel that way; we certainly don’t want to have everyone abandon the clarity and ease of modular code and start putting everything in-line in huge, long functions. But for applications where you are concerned about security, be aware that users can erase sub-functions and replace them with their own functions.

Once they do that, all bets are off. If your main calling routine now calls their function (because its name matches the one that you used to use), they can, for instance, read the values of the variables that you have localized in your main calling routine. They may be able to slip past your security checks, and chances are, you’ll never know that they did it. While single “stand-alone” functions are usually discouraged, this is one case where they are needed.

To re-emphasize the point: in any application in which you are concerned about the security aspects of what a user may access with your functions, the only way to ensure that he cannot tamper with the innards of your code is to package everything in one single stand-alone function, with no sub-functions and no global variables. Sorry. If you don’t do that, you’re kidding yourself.

But hold on, before you re-write everything; fortunately, there is an easier way. Read about our “LOCALIZE” functions on page 59.
They can do the job of localizing the sub-functions for you, and still let you maintain good building-block functions.

**Prompting for Input from the User**

If you need to prompt the user for some data while he is using your function, you should always use “!"-input instead of ""-input. You should make certain that ""-input is never used within a locked function either directly or within any function that is called by that locked function. You should avoid ""-input because the user may otherwise be given the opportunity to gain too much information about the function—even though it's locked—by using the State Indicator commands [""SI"" and "]SINL"] when the keyboard is unlocked awaiting their input. Besides maintaining the security of the locked function, the use of ""-input also permits the use of more versatile and powerful input routines than does ""-input.

To reiterate that warning, do not use ""-input in a sensitive application! If you do, it's not secure. Refer to the discussions of Dyadic Execute on pages 66-71 for additional thoughts regarding techniques for prompting for user input safely.

**Is This Really the Right Approach?**

Locked functions are fine for cases where you want to keep the "browsers" out of your code (such as places where you may be using a new algorithm which you plan to publish). But don't consider locking to be absolute. If the workspace is transferred to another (non-APLSV) system, it is possible for a knowledgeable programmer to unlock the functions. And future implementations of APL (such as APL2) may treat the whole concept of locking functions differently than we do now. If you have particular needs or concerns regarding locked functions, we would welcome the chance to discuss it with you. Call us on the APL Hotline, T/L 695-1234, and ask for some programming assistance.

For places where you need to protect data (as opposed to APL code), a much better approach is to put the data into a "reserved" TSIO dataset, and use a "command dataset" to control access to that data. In this way, the data would not appear in the workspace at all unless the user had validated access authority. Refer to the discussions of "Dataset Privacy" on pages 85-91.

---

Bumper sticker of the week: ![FNS] (seen on an "AUDI-T".)
Creating Functions Dynamically (\texttt{\textcopyright FX})

The definition of a function can be established, or "fixed," by applying the system function \texttt{\textcopyright FX} to its character representation. The function \texttt{\textcopyright FX} produces as an explicit result the character vector which represents the name of the function being fixed, while replacing any existing definition of a function with the same name. A halted function may be replaced using \texttt{\textcopyright FX}, but this will not replace the active copy on the stack.

The fix function has both a monadic and a dyadic form. We'll look at the monadic case first; the dyadic case is covered on page 57.

An expression of the form \texttt{\textcopyright FX M} will establish a function if both of the following conditions are met:

1. \texttt{M} is a valid representation of a function. It may be a matrix (with each row representing a function line). Any matrix which differs from the canonical representation only in the addition of non-significant spaces (other than rows consisting of spaces only) is a valid representation.

2. The name of the function to be established does not conflict with an existing use of the name for a variable, a label, or a group.

The first row of the matrix represents the function header and must be a format appropriate to a function header. The remaining rows, if any, constitute the function body, and may be composed of any sequence of valid APL characters.

If the expression fails to establish a function then no change occurs in the workspace and the expression returns a scalar index of the row in the matrix argument where the first fault was found. If multiple errors are present, only the first will be reported. (This value is origin dependent.) If the argument of \texttt{\textcopyright FX} is not a matrix, a \texttt{RANK ERROR} will be reported, and if it is not a character array, a \texttt{DOMAIN ERROR} will result.
For example,

\[ M \leftarrow 2 \ 8 \ \rho \ 'Z\text{+ROOT} N Z+N*0.5' \]

\[ M \]
\[ Z\text{+ROOT} N \]
\[ Z+N*0.5 \]
\[ \square FX M \]

\( \text{ROOT} \)

\( \text{\textit{The name of the function is returned if } } \square FX \text{ completed successfully} \)

\[ \sqrt{\text{ROOT}}[[]] \sqrt{\text{Z\text{+ROOT} N}} \]
\[ \sqrt{\text{Z+N*0.5}} \]
\[ [1] \]
\[ [1] \]
\[ \square FX 2\ 1\ \rho \ 'Z+N \text{ ROOT} Z Z+A*7N' \]

\( \text{ROOT} \)

\( \sqrt{\text{ROOT}}[[]] \sqrt{\text{Z+N ROOT A}} \)

\( \sqrt{\text{Z+A*7N}} \]

\[ [1] \]

\[ [1] \]
\[ \square FX 1\ 2\ 3 \]
\[ \text{RANK ERROR} \]
\[ \square FX 1\ 2\ 3 \]
\[ \wedge \]
\[ \square FX 2\ 1\ \rho \ 'Z+A*7N' \]
\[ Z+N \text{ ROOT} Z' \]

The first row of the matrix isn’t valid as a header line, so the function can’t be created.

Creating Locked Functions Dynamically (Dyadic \( \square FX \))

Now let’s look at the dyadic form of \( \square FX \).

A left argument may optionally be supplied to \( \square FX \) to control the optional locking of the newly-created function. If present, the argument must be a boolean value. If there is no left argument, the action will be the same as 0 \( \square FX M \). A 1 specifies that the resultant function is to be locked, a 0 specifies that it’s to be unlocked.

\[ \square FX M \]

...resulting function will be unlocked

0 \( \square FX M \)

...resulting function will be unlocked

1 \( \square FX M \)

...resulting function will be locked

The ability to create locked functions under program control is important for such applications as bringing proprietary APL functions in from a TSIO file.
Local Functions

`OFx` may be used to create local functions within other functions. The local names on the header line of a function don't only refer to variables; they may also represent functions. By placing the name of the function to be localized on the header line of the calling function, and using `OFx` to create the new function, a function may be localized just as easily as any variable.

This is a good technique to use when you want to use a small piece of code many times within your main function (an obvious place for a sub-function), but you want to make sure that nobody can simply erase a global sub-function and replace it with a copy of their own, which does who-knows-what.

An example of a small function that often falls into that sort of category is an error-checking function for checking the return codes from TSIO. It would typically get used *many* times in a single main calling routine. We don't want to have to code all of the occurrences of that code in-line... that's a lot of work, and it creates maintenance problems. And we also don't want to leave the function out where it can be tampered with. So we do the same thing with the function that we do with any variable in the same situation: we localize it.

Let's look at an example of a function with a localized sub-function. When we give this function to other people to use, we plan to lock it, and we want to protect its sub-functions, too. Let's assume that our users are not authorized to see all of the data in the file... part of the purpose of this `SALES` function is to do the data-reduction side-step our protection. Here's how we can ensure that, by localizing our error-checking function called "ERR":

```
       v Z+SALES;CTLSALES;A;M;IO;ERR
[1]   RETURNS CURRENT DATA FROM THE SALES FILE
[2]   IO+1
[3]   A+370 SVC 'CTLSALES'
[5]   LOCALIZE THE TSIO ERROR-CHECKING FUNCTION, TO PREVENT TAMPERING:
[7]   M=M,[1] 2 20p'=((1=pR) a b R+R)O+O' 'ERROR CODE: ' R'
[8]   A+1 OFX M ←This creates the local function
[9]   CTLSALES+'
[10]  ERR CTLSALES
[12]  CTLSALES+IC DSN='123456 SALES.ACCESS(9)'
[13]  ERR CTLSALES
[14]  The rest of the function goes here. By the way, you would want that TSIO return-code checking function to be somewhat more sophisticated than this for a real application. Here, we are simply showing the technique of function localization.]
    * Once we lock this main function, the sub-function is also protected.
```
Localizing Functions Dynamically

Workspace 10 LOCALIZE is designed to let a user make a copy of global functions and variables to appear as local functions and variables within a function of your choice. This provides an easy-to-invoke security feature, preventing users of your application from tampering with these sub-functions or variables.

The main function in the workspace is "LOCALIZE". Its form is 'FN' LOCALIZE LIST, where FN is the name of the function that is to receive the local functions, and LIST is a list of objects to be localized (as a matrix of one name per line, or a vector of names separated by blanks, commas, or semicolons). The explicit result from "LOCALIZE" is a character matrix showing the lines which have been inserted into "FN" (for reference).

If the left argument (FN) is elided, then no function is rebuilt, but the explicit result still returns the data that can be inserted manually into a function.

Examples:

)LOAD MYWORK
SAVED ....

)PCOPY 10 LOCALIZE LOCALIZEGP
SAVED ....

 v FOO;A
 [1] A+0
 [2] 
 [3] 'chk try 'SR DSN+....
 v

Z+'FOO' LOCALIZE 'CHK TRY QLE Q'
4 OBJECTS TO BE LOCALIZED...
NAME OF ERROR LABEL WITHIN YOUR FN TO BE BRANCHED TO IF ERRORS OCCUR: 'O'
FOO

 v FOO;A;L;FX;CHK;TRY;QLE;Q
 [1] A+0
 [2] L+1
 [3] 'CHK TRY 'SR DSN+....
 v

In this example, the function "FOO" has been modified such that the functions called "CHK" and "TRY" and the global variables called "QLE" and "Q" have been localized within "FOO". After the conversion, they still exist as global objects, and should be )ERASEd before the application is made available to other users.
Notice that the original function had four lamp symbols ("MMMM") on line two; this was to indicate to the "LOCALIZE" function just where the definitions of the new local objects should appear. After the conversion, these symbols appear both before and after the newly localized objects (lines two and nine).

Line three is a local variable that's used to indicate whether the local functions (if any) are to be created as locked functions (L+1) or as unlocked functions (L+0). The default is "L+1"... "L+0" tends to circumvent the security of the application, and should be used for initial debugging only (if indeed at all).

Line four is the definition of a local function called "FX"; it will appear any time that any other functions are localized, and is used for "boot-strapping" the other functions in. Since the other functions appear in a somewhat compressed form (to help to prevent WS FULL problems), "FX" is used to rebuild them when you run your function.

On lines five through eight are the definitions of the four objects that we asked to localize.

If you wish to edit the localized functions, it's not practical to edit them "in place" after they have been localized, so instead, their definitions may be removed from your function. You would then copy the original global functions and variables into your workspace, make whatever changes that you want, and re-localize the objects. To remove the local functions, and put the "FOO" function back to its original state:

```
UNLOCALIZE 'FOO'

FOO
   v FOO;A
[1]   A+0
[2]   MMMM
[3]   ' CHK TRY 'SR DSN+....
   v
```

Limitations

During the execution of your function (the ubiquitous "FOO", in this example), there will be two copies of each object: the active local object, and the one-line definition that created it. Therefore, considerable extra space may be needed to use this procedure. If your application is already battling a WS FULL problem, this procedure will only aggravate it [and you].

Realize also that it takes time to rebuild each of the local objects every time that you run your function. This procedure is therefore not without a price, but it may add an extra measure of security to your application that would be otherwise unavailable. It will certainly keep the sub-functions from prying eyes and tampering hands.
The "SETBOMB" Workspace

Workspace 10 SETBOMB is designed to help you to set up checks within your existing functions to ensure that only authorized users can execute your code. This is an easy way to make your sensitive workspaces more secure. If an unauthorized user attempts to execute it, the functions will "bomb"-out, by expunging all of the other functions and variables in the workspace and preventing that user from continuing.

The salient function in this workspace is simply called "SETBOMB". It prompts you through the choices in easy-to-follow terms, and then automatically inserts the selected patches into your functions.

To use this facility:

`LOAD your workspace`  
`WSID SECUREWS:LOCKWORD`  
`COPY 10 SETBOMB SETBOMB`  
`Execute "SETBOMB" once for each function to be bombed`  
`ERASE SETBOMB`  
`Lock all functions`  
`SAVE`

The "SETBOMB" function generates its "bombs" and inserts them into any specified functions, to check any or all of:

- the user's sign-on
- the latent expression
- the date
- any other logical condition that you specify.

When executed, it prompts you for:

- name of function to be bombed
- place where patch is to be inserted
- sign-ons of authorized users
- latent expression
- an expiry date
- any other condition that you want to have checked
- message to print when bomb is triggered
- should bomb expunge variables, functions, or both

For the most protection, you should have at least two bombs in each workspace:

1. a latent function checking (at least) the user's sign-on;
2. a master function checking (at least) the user's sign-on and the latent expression

This will protect you from all unauthorized use except someone using your own sign-on. However, you can take care of that by requiring the user to set some kind of variable password after loading the workspace, and setting a bomb to check it.

In using "SETBOMB", be sure to keep an unbombed, separately-named copy of your workspace. The bombs work all too well; you can easily lose the workspace if you trigger one accidentally.
Event Handling

While we always try to “do things right the first time” (don’t we?), there comes that inevitable time when we make a mistake. APL has planned ahead for this eventuality and lets you provide a layer of “blanket coverage” against errors.

“I claim not to have controlled events, but confess plainly that events have controlled me.”

—Abraham Lincoln, in a letter to A. G. Hodges.
4 April 1864

Sure, every APL programmer has been in this situation. Clearly, we need to let our functions control events. But before we learn how to handle those events, perhaps we should ask, “What is an event?” Fair enough. Webster defines an “event” as this:

event \i·'vent\ n 1: the fundamental entity of observed physical reality represented by a point designated by three coordinates of place and one of time in the space-time continuum postulated by the theory of relativity 2: something that happens

—Webster’s New Collegiate Dictionary, 1974

[Hmmm, let’s use the latter definition here... ] Okay, so an event is just something that happens. But surely we don’t wish to invoke special handling for everything that happens; we want to be a bit selective. People tend not to observe the day after a birthday with quite the same zeal as the birthday itself. And in the APL environment, the same situation holds. Any action in APL can be considered to be an event; assignments, branches, calling functions are all events. ...They just may not be noteworthy events. So what makes an event noteworthy? ...The desire to observe it.

There are many places in programming where foreseeing some situation may not be possible, or even if it’s possible, may not be practical to measure. Take, for instance, the most common example of checking for an error. Your programs, of course, should do “extensive error checking” if they are to be used in a production environment; anyone will tell you that.

But some things just aren’t practical to check. You are always faced with a potentially infinite number of things that could happen during the execution of your programs—all things which are outside of normal operations. What happens if there’s an unexpected domain error? What happens if an error occurs that I haven’t thought of? What happens if the user presses the ATTENTION (or PAZ) key and interrupts the execution part way through? ...Welcome to Event Handling.

“Once the toothpaste is out of the tube, it’s hard to get it back in!”

—H. R. Haldeman
APL provides the means for letting execution simply run its course, and if an abnormal situation occurs, for helping you to take corrective action.

You may have noticed by now that the previous page spoke of events as being a broad subject, but that we are now referring mostly to errors. Errors are indeed only one type of event that we might want to handle, but errors also happen to be the one type of event that the most people have had the greatest interest in handling. Because of this, most of the event handling in APL is aimed toward the handling of errors. So errors aren't the only things that are considered to be “events” (controlling the use of the **ATTENTION** key was one example of another kind of event) but errors will be the main subject of our following discussion.

*Take the Worry Out of Running Code*

If you are a programmer who writes any applications that you give to non-programmers to run, having one final catch-all procedure for trapping a totally-unforeseen bug before it halts the function and prints out a line of “Greek” (as non-APLers always seem to call it) can perhaps let you get a full night’s sleep for a change. This will give you the facility, for instance, of creating an entry in a logging file that you may wish to construct if a bug occurs anywhere in your code while any of your users are running it. Okay, we know, your programs haven't had any bugs in them for years. Sure. But—even if we did believe you—and we do—yeah, sure. even things that work reliably for years will still fail eventually; everything does. [Escalators are real reliable, too, but I can still remember being trapped one day for four hours on an escalator when the power went off. ...You don’t forget those things.]

In our discussions of event handling, we should take a look at the “execute” function. Execute isn’t properly an event handling function, but its action is so similar to that of APLSV’s event-handling function—"dyadic execute"—that monadic execute just seems like required reading for this context. So, here goes.

---

**McGurk’s Law:**

"Any improbable event which would create maximum confusion if it did occur, will occur."

—H. S. Kindler.

from “Organizing the Technical Conference.”

Reinhold Publishing Company, 1960
Execute

Any character vector or scalar can be regarded as a representation of an APL statement (which may or may not be well-formed). The monadic function denoted by "." takes as its argument a character vector or scalar and evaluates or executes the APL statement it represents. When applied to a character array that might be construed as a system command or the opening of function definition, an error will necessarily result when evaluation is attempted, because neither of these is a well-formed APL statement.

The execute function may appear anywhere in a statement, but it will successfully evaluate only valid (complete) expressions, and its result must be at least syntactically acceptable to its context. Thus, execute applied to a vector that is empty, contains only spaces, or starts with "-" (branch symbol) or ~ (comment symbol) produces no explicit result and therefore can be used only on the extreme left. For example:

```
Z←''
VALUE ERROR
Z←''
∧
```

The domain of . is any character array of rank less than two, and RANK and DOMAIN errors are reported in the usual way:

```
C←'3 4'
+/Z C
```

```
RANK ERROR
∧
```

```
C←'3 4'
+/Z C
```

```
DOMAIN ERROR
∧
```

An error can also occur in the attempted execution of the APL expression represented by the argument of .; such an indirect error is reported by the error type and followed by the character string and the caret marking the point of difficulty. For example:

```
'5+0'
DOMAIN ERROR
5+0
∧
```

```
'WSID'
VALUE ERROR
') WSID
∧
```

7 There's an additional restriction that only rarely surfaces: the character string must be composed only of valid APL characters. For example, even though the APL system uses various characters to control the operation of the terminal (such as idle characters, linefeed characters, and end-of-block characters), these characters are not APL characters, and therefore, they may not be used within a string that is to be operated upon with the execute function.
Using Execute to Assign a Value to a Supplied Name

An example of the use of Execute is a situation in which the user of the application is supplying a name for a variable, which then needs to have data assigned to it. The problem is to find a way to get data stored into a name which is really just represented as a character string. So that you don’t lose sleep over this one, we’ll just show you how it’s done:

```apl
DATA 3 5 7 11 13 17 19 23
NAME MYDATA
VALUE ERROR MYDATA
\pNAME 6
\sNAME,' +DATA'
MYDATA 3 5 7 11 13 17 19 23
```

Here’s some existing data

We want to move it into a variable having this name

which doesn’t currently exist.

This will do it

Here’s the new variable

Now, if you wish to extend it, and catenate more data to what’s already there, you can do this:

```apl
\sNAME,' +\NAME',29 31 37 41'
or this:
\sNAME,' +(\NAME),29 31 37 41'
```

Store under this name...

The data that previously existed under the same name...

...Followed by this new data

```
MYDATA 3 5 7 11 13 17 19 23 29 31 37 41
```
Dyadic Execute

Discussions of execute have often alluded to the idea that \( \mathbf{\oplus} \) can be used as "an alternative to \( \mathbf{R} \)-input offering more progam control." Well, maybe, but any of you who have tried to actually do this have probably discovered that the problems start when the first user of your application types in an entry that's not a "well-formed APL expression":

\[
\mathbf{\oplus} \mathbf{~}
\]

\[
\begin{align*}
2+
\mathbf{SYNTAX\ ERROR}
2+ \\
\wedge
\end{align*}
\]

A function that is trying to prompt for a character string that represents a vector of floating-point numbers, and then execute it to get the string into numeric form, may well spend most of its time simply ensuring that the execution is going to work; every possibility of an erroneous input must first be checked. Perhaps the classic example of this is using \( \mathbf{\oplus} \) to invert a matrix: there are rules governing the acceptability of the matrix for inversion, but checking the matrix will probably take longer than the inversion. A nice approach would be to simply try it, and back off if it fails. Normal error behavior involves a halt to execution if it fails. But sometimes it's undesirable for an application to stop. Some means of getting control when an error occurs is needed. This may easily be done with dyadic execute.

Consider the case of \( \mathbf{\oplus} \mathbf{R} \). If \( \mathbf{R} \) can't be executed, an error message will be returned (such as \( \mathbf{SYNTAX\ ERROR}, \mathbf{LENGTH\ ERROR}, \mathbf{WS\ FULL}, \) and so forth). Dyadic execute, \( \mathbf{L} \mathbf{\oplus} \mathbf{R} \), will return exactly the same result as \( \mathbf{\oplus} \mathbf{R} \) if the execution is successful; the left argument will be ignored. But if \( \mathbf{R} \) can't be executed, the expression will be treated just as if it was \( \mathbf{\oplus} \mathbf{L} \). In particular, if the left and right arguments are both invalid, an error message will be reported that will look just as if the expression had been \( \mathbf{\oplus} \mathbf{L} \):

\[
\begin{align*}
\mathbf{\oplus} '3+'
\mathbf{SYNTAX\ ERROR}
3+
\wedge
\end{align*}
\]

\[
\begin{align*}
2+2 '3+' \\
4
\end{align*}
\]

\[
\begin{align*}
'2+2 '3+'
\mathbf{SYNTAX\ ERROR}
2+ \\
\wedge
\end{align*}
\]
A particularly useful application of dyadic execute is "'→OOPS' 'FOO'", in which any problem in the character string FOO which would prevent it from being executed will cause a branch to label OOPS.

Dyadic Execute will switch arguments after any occurrence of an error in the right argument, regardless of the depth of the function calls that may have occurred in the right argument. For example, consider "'→OOPS' 'FN' ", where FN is a function. If FN calls another function, FN2, which subsequently encounters a DOMAIN ERROR, the error will not be reported, but rather, ≜ will immediately abandon execution of the right argument, and instead will execute the left argument ('→OOPS').

Be aware of a frequent trap: a common approach is to enter an expression such as "Z+'→OOPS' 'FOO'" with the idea that an error would cause the function to branch. ...taint so, McGee. If the input is executable, the expression can be viewed as "Z+'FOO'". But if it's not executable, the expression becomes "Z+'→OOPS' ", or "Z+'→OOPS' ", an immediate error. Therefore, although it's longer, a bit slower, and somewhat more cumbersome, what's really needed is "'→OOPS' 'Z+' 'FOO'".

Please realize that this primitive is not meant to be an all-encompassing coverage of generalized error side-tracking (...you'll have to wait for us to install APL2 in order to enjoy that benefit). [For instance, you may notice that it's not possible for APLSV to find out what the error was(!) ...c'est la vie.] There are many situations where recovering from an error during execution will not be possible. But for situations in which you can anticipate a specific problem, and have a remedy for it, dyadic execute may be just the ticket.

If you do wish to pursue the applications of dyadic execute in the new APL2 systems, realize that this primitive has been slightly re-packaged for that environment. It is now called "DEA" (=execute alternate) under VSAPL and APL2; it's functionally and operationally the same as dyadic execute on APLSV.
Here’s an example of a simple input-checking function which will prompt for numeric data, and will re-prompt if the input can’t be executed:

```
v Z+NUM T
[1] ←PROMPTS USER WITH MSG IN RIGHT ARG; EXECUTES INPUT
[2] START:T←T
[3] Z←"T
[4] →(Zewart="')/EXIT
[5] →OOPS' & 'Z=.1Z'
[6] →0
[7] OOPS:←'INVALID, PLEASE RETRY...'
[8] →START
[9] EXIT:Z←0

R←NUM 'ENTER NUMERIC STRING: '
ENTER NUMERIC STRING: 1 2 3 4.5.6
INVALID, PLEASE RETRY...
ENTER NUMERIC STRING: "3.7"
INVALID, PLEASE RETRY...
ENTER NUMERIC STRING: 1 2 3 4. 5.5

0
```

Section 5: Protecting Your APL Functions
Note that if the function were “simplified” a bit, it could become difficult for a well-meaning user to exit from the function:

```
   v Z+NUM2 T
[1] # PROMPTS USER WITH MSG IN RIGHT ARG; EXECUTES INPUT
[2] START: [T-]
[3] 'OOPS' ['Z+', [T]]
[4] 0
[6] ->START
```

(The user calls the function, but then decides to cancel or interrupt the function)

```
R+NUM2 'ENTER NUMERIC STRING: '
ENTER NUMERIC STRING: [user just presses ENTER]
INVALID, PLEASE RETRY...
ENTER NUMERIC STRING: [user depresses the INTERRUPT key, trying to halt the execution, but to no avail.]
INVALID, PLEASE RETRY...
ENTER NUMERIC STRING: 

(...and on, ad infinitum...)
```

...The Moral: Although there may be some legitimate times where you want to “trap” a user’s input without letting him interrupt the function, be sure that you use this sort of capability with discretion; don’t make your functions unresponsive to the user.

Be aware that, using dyadic execute, it is possible to write uninterruptible functions. You need to carefully consider the implications of a bug in your code, which may prevent a user from interrupting the execution of your code. If it can’t be interrupted, and continues to loop at high speed, your users may run up a very large bill in a very short time. And guess who they’ll come back to with their bill.... Be careful that you don’t work yourself into a box. This one could cost you elephant bucks.

Also take care to avoid name conflicts in functions that use either monadic or dyadic execute. A user who is entering lots of repetitive data may wish to set up a variable in the workspace, and enter its name in response to the prompt for input. That’s fine, but with this particular function he would suddenly discover “mysterious” operations occurring if the name that he chose was “T” or “Z” or “START” or “OOPS”.

Controlling the Security of Applications on Kingston APLSV —The APL Jot® Dot Times—
If you are interested in reading more about the topic of Event Handling, you can order a copy of An Introduction to APL2 (SH20-9229). This manual has a chapter on Event Handling as it pertains to APL2, and will show the direction that you can consider taking with the design of your applications as soon as Kingston APL converts to APL2.

Alan Graham discussed "Examples of Event Handling in APL2" at the APL83 Conference in Washington, D.C. This discussion shows many of the features that are available under APL2, the system to which we will be converting over the course of the next couple of years. Many of the IBM site libraries have the proceedings from this conference, published as the Volume 13, Number 3 issue of the "APL Quote Quad". The proceedings are also available from ACM (The Association for Computing Machinery); order number 554830.
After reading the section on dyadic execute, I was surprised that its usefulness was not demonstrated in the section on ambi-valent functions. Your example of the \textit{ROOT} function could very easily have employed dyadic execute:

\begin{verbatim}
\texttt{v Z+N ROOT A}
\texttt{[1] 'Z+A*2' = 'Z+A*N'}
\texttt{V}
\end{verbatim}

\texttt{NOT recommended!}

---

We received several such statements as a result of a previous Kingston APL newsletter. Say what you will about programming style, but we feel that this sample function brings up a potentially dangerous situation. You are anticipating a \texttt{VALUE ERROR} if \texttt{N} isn't assigned, forcing execution of the left argument. However, any error arising from the use of the left argument will have the same effect---even \texttt{WS FULL}.

As an example of this, \texttt{"3 ROOT 64 729 4096"} correctly finds the cube root of each of the values, yielding a result of \texttt{"4 9 16"}. So far, so good. But realize that \texttt{"2 3 ROOT 64 729 4096"} \textit{should} produce a \texttt{LENGTH ERROR}; instead it returns \texttt{"8 27 64"}---the \texttt{wrong answer}---and would allow a calling function to continue.

A more proper way of checking for the existence of the left argument is by the use of \texttt{ONC}. \texttt{"ONC 'N'"} (in this example) will return a \texttt{2} (meaning "variable") \textit{only} if a left argument was supplied; it returns a \texttt{0} if the \texttt{ROOT} function was called monadically.

We recommend against the use of dyadic execute to circumvent normal, quick checks like this that have been traditional in the past, and we especially recommend against it if it's used as a return to "one-liners."
Section 6: Protecting Your TSIO Datasets

What Datasets Do You Own?

If you are like the rest of us, you probably own a couple of datasets that you no longer need, or didn’t even know existed. How do you find out? (...thought you’d never ask).... Simply:

```
)LOAD 1 FILELIB
```

and type “STATUS FILES”. Your dataset names will be displayed along with additional information such as the number of tracks allocated, the percentage utilized, when the file was created, and when it was last used. (“CHAR FILES” also provides some useful information about the characteristics of those files.)

You can think of “FILELIB” as being a bit like the “)LIB” command... it shows you your “library” of file names.

The functions in workspace “1 FILELIB” allow APL users to inquire about their OS datasets as used through TSIO. Information is available in four general areas:

- the names of datasets
- the status of datasets
- the characteristics of datasets
- the IBM security classifications of datasets
General Inquiry Form

(choose 1) (choose 1)

| SORTBYSIZE | STATUS     |
| TOTAL      | (selection— see below) |
| SORTBYPACK | USAGE      |
| SORTBYDATE | CHAR       |
| SORTBYDATE | (selection— see below) |
| SORTBYDATE | (selection— see below) |
| SORTBYCLASS| (selection— see below) |

...where (selection) is any one of these:

... 'XYZ' (beginning of file name)
... FILES
... FILES ON 'APL501 APL502' (pack names)
... FILES ON TEMPPACKS
... FILES ON TAPE
... FILES ON SYSTEM
... FILES ON SYSTEM EXCEPT TEMPPACKS
... FILES ON SYSTEM EXCEPT 'APL501 APL502'
... FILES CLASS 'IUO'
(...etc.)

Typical Inquiries

CLASSIFY FILES
STATUS FILES
STATUS 'XX'
TOTAL FILES
USAGE FILES
SORTBYSIZE STATUS FILES
SORTBYDATE STATUS FILES
SORTBYDATE STATUS FILES
TOTAL SORTBYDATE STATUS 'XYZ'
CLASSIFY FILES ON 'APL501 APL502'
TOTAL STATUS FILES ON 'APL501 APL502'
TOTAL STATUS FILES ON TEMPPACKS
TOTAL STATUS FILES ON TAPE
TOTAL STATUS FILES CLASS 'IUO'
GROUPS
RENAME
DELETE

CLASSIFY 'XX'
STATUS IC FILES
CHAR FILES
TOTAL STATUS FILES
USAGE FILES ON 'APL501'
SORTBYPACK STATUS FILES
SORTBYDATE STATUS FILES
SORTBYCLASS STATUS FILES

Section 6: Protecting Your TSIO Datasets
Here are some samples of usage of some of the major functions:

**R-STATUS dsnames**

Takes a dsname or names as the right argument and returns a matrix of status information about the datasets belonging to the user and included in the right argument. The display looks like this:

```
STATUS FILES
[3]
VOLUME CREATED LAST-USE TRACKS PCT EXT SEC RECFM DATASET-NAME
APL523 11/05/81 06/20/83 34 97 1 F 123456 DOCUMENT
APL501 12/11/81 06/21/83 1100 1 F 123456 REPORT
---ARCHIVED----- 11/22/78 325 59 1 F 123456 SALES
```

Here's what the various fields represent:

- **VOLUME** — the volume identification of the disk pack containing the dataset.
- **CREATED** — the date (month/day/year) that the dataset was originally formed.
- **LAST USE** — the date (month/day/year) that the dataset was last accessed (read or written) by anyone.
- **TRACKS** — the number of tracks of disk space allocated to the dataset.
- **PCT** — the percentage of allocated tracks actually being used in the dataset.
- **EXT** — the number of contiguous areas (extents) on the disk used to contain the dataset.
- **SEC** — the number of blocks (physical records) requested as secondary allocation space when the dataset was created. For more information, see the description of the space parameter in The APLSV Version 3 User's Guide (SH20-9087).
- **RECFM** — the record format of the dataset, indicating whether the data is fixed or blocked.
- **DATASET-NAME** — the name of the dataset, with a negative number indicating that the dataset is "reserved" (that is, restricted to access by you only). See pages 85-91 for a complete discussion of this.
- **CLASS** — the security classification of the dataset, (only shown when SORTBYCLASS is specified).
**USAGE dsnames**

This function is similar to *STATUS*. Since its display is slightly fancier than the output of *STATUS*, it takes a little longer to run. *USAGE* always automatically totals the information. The security classification is always shown when *USAGE* is run.

### USAGE FILES

<table>
<thead>
<tr>
<th>VOLUME</th>
<th>CREATED</th>
<th>LAST-USED</th>
<th>TRACKS</th>
<th>PCT</th>
<th>EXT</th>
<th>CLASS</th>
<th>DATASET-NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>APL523</td>
<td>THU 05 NOV 81</td>
<td>MON 20 JUN 83</td>
<td>34</td>
<td>97</td>
<td>1</td>
<td>IC</td>
<td>123456 DOCU</td>
</tr>
<tr>
<td>APL501</td>
<td>FRI 11 DEC 81</td>
<td>TUE 21 JUN 83</td>
<td>1</td>
<td>100</td>
<td>1</td>
<td>UNCL</td>
<td>123456 REPO</td>
</tr>
<tr>
<td>(ARCHIVED TO TAPE)</td>
<td>WED 22 NOV 78</td>
<td>325</td>
<td>59</td>
<td>1</td>
<td>IUO</td>
<td>123456 SALE</td>
<td></td>
</tr>
</tbody>
</table>

---

**TEMPORARY ON-LINE:**

- 0 FILES
- 0 TRACKS

**PERMANENT ON-LINE:**

- 2 FILES
- 35 TRACKS ON DISK

**TOTAL ON-LINE:**

- 2 FILES
- 35 TRACKS ON DISK

**OFF-LINE:**

- 1 FILE
- 325 TRACKS ARCHIVED

**TOTAL:**

- 3 FILES
- 360 TRACKS TOTAL

---

**R+CHAR dsnames**

Takes a name or names as right argument and returns a matrix of expressions which are subsets of the expressions which might have been used to allocate the respective datasets.

```
CHAR 'SALES'
RECFM=F,BLKSIZE=19069,LRECL=10969,SPACE=(19069,(325,0)),DSN=....
```

Note that "STATUS", "CLASSIFY", "USAGE", and "CHAR" can take "FILES" as their right argument. Some examples of their uses are:

- **CLASSIFY FILES**
  - classify all files
- **CHAR FILES**
  - characteristics of all of the files
- **STATUS 'ETC'**
  - status of all files starting with "ETC"
- **STATUS 'ETC'**
  - status of the single file named "ETC" (not just those starting with "ETC"... notice the blank after the name)
- **USAGE FILES**
  - status and totals combined
- **TOTAL STATUS FILES**
  - status and totals combined
- **TOTAL FILES**
  - just the total information

---

Section 6: Protecting Your TSIO Datasets
One of the more useful inquiries is simply "TOTAL FILES", which will show how many files you have, and how much space you are being billed for:

<table>
<thead>
<tr>
<th>ON-LINE:</th>
<th>1 FILE</th>
<th>1 TRACK ON TEMPORARY DISK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON-LINE:</td>
<td>1 FILE</td>
<td>1 TRACK ON PERMANENT DISK</td>
</tr>
<tr>
<td>ON-LINE:</td>
<td>2 FILES</td>
<td>2 TRACKS ON DISK</td>
</tr>
<tr>
<td>OFF-LINE:</td>
<td>2 FILES</td>
<td>650 TRACKS ARCHIVED ONTO TAPE</td>
</tr>
<tr>
<td>TOTAL:</td>
<td>4 FILES</td>
<td>652 TRACKS TOTAL</td>
</tr>
</tbody>
</table>

**Notes on Temporary versus Permanent Datasets**

Permanent TSIO datasets are created by using the "ALLOCATE" function in workspace "AIDS". Datasets created by any other means are temporary, and are deleted by the system one week after they are created. For more information regarding the use of "ALLOCATE", see page 79.

**How and Why Datasets Are Archived**

Due to the demands for APL disk space available for new TSIO file allocations, we have installed a data set archiving system. Files that have not been accessed for six months are eligible for archiving. Likewise, files that belong to accounts which have been deleted from the system ("orphan files") are also eligible for archiving. All of these data sets are dumped to tape. One copy is kept in our library for future restores, and the other is stored onsite as a backup copy. The archive tapes will be retained for four years. Once a file has been archived, the owner will no longer be billed for its space. (Note: file archiving by user request is not available.)

If you try to open a TSIO data set that has been archived, you may get a return code 18. The old description of TSIO return code 18 was "VOLUME UNAVAILABLE". The message has been changed to "OFFLINE/ARCHIVED". If you use the "QLE" matrix for TSIO error reporting, you can copy an updated version from workspace 30 TSIO.

Status information for your archived files (in addition to on-line data sets) will still be available in workspace 1 FILELIB.

**Deleting an Archived file from the listing:**

Any file, archived or on-line, may be deleted by using the "DELETE" function in 1 FILELIB... but there are some notes of caution that you should be aware of, so be sure to read the complete discussion on page 83.

The programmers amongst you may wish to use the "ARCHDEL" function to delete archived files. The "ARCHDEL" function takes
as a right argument a single file name in quotes or a matrix of file
names (one per line) to be removed from your FILELIB listing.
This change will not be reflected, however, until the next time
that the FILELIB dataset is updated (usually early each morning).

Once a file is removed from archive it is gone for good, so please
use care when deleting your files.

Getting Data Back from Archives

To get an archived file restored, call The APL Control Centre on
T/L 695-6772. The files will be recalled from archive as quickly
as possible, usually within the same day.

Miscellaneous Additional FILELIB Functions

RENAME Renames a specified file
DELETE Deletes a specified file
GROUPS Displays a list of the first-level qualifiers and the
number of files which begin with that qualifier

Files that have been archived are displayed with “ARCHIVED”
under the volume heading in the status display. To selectively
display them, type:

STATUS FILES ON TAPE

To display only your on-line data sets, type:

STATUS FILES ON SYSTEM

To exclude temporary files from the display, you could say:

STATUS FILES ON SYSTEM EXCEPT TEMPPACKS

You can selectively display your files on any pack, or exclude
specific volumes from the list like this:

STATUS FILES ON 'APL501 APL502'
CHAR FILES ON SYSTEM EXCEPT 'APL501 APL502'

You can selectively display your files according to security class:

STATUS FILES CLASS 'UNCL'

To display those files which have not yet been classified at all,
enter:

STATUS FILES CLASS ''

To obtain a list of first level qualifiers and the number of files
that occur under each, type:

GROUPS

To display only your IC datasets, type:

STATUS IC FILES

For a discussion of the implications of deleting a dataset, please
refer to page 83.

Further reading is available from The APLSV Version 3 User’s
Guide (SH20-9087). Refer to Chapter 3.
Classifying Your TSIO Datasets

Corporate Security Instruction #104A states that “Ownership [of data] conveys authority and responsibility for ... classifying the assets and reviewing control and classification decisions” [page 133]. Corporate further requires that “Owners must ... classify input, output, data, and software; review classification and controls for appropriateness and adequacy at least annually ...” [page 148], and further that “all data and software must be associated with its owner, classified appropriately and controlled accordingly.” [page 149]. To be in compliance with all of that, all of your permanent TSIO datasets must be classified. But don’t panic; classifying them is really quite simple. Here’s how it’s done.

Classifying Newly-Created Datasets

New datasets are created by using the “ALLOCATE” function in workspace “i AIDS”. That function now also prompts for the dataset’s security classification, so newly-created datasets will be classified as they are created.
Here's a sample session, showing the creation of a new TSIO dataset, and specifying a security classification of Internal Use Only for it:

```
)LOAD 1 AIDS
SAVED 10.22.11 08/24/84
ALLOCATE

TYPE ? FOR ASSISTANCE

====> DO YOU WISH THIS DATASET TO BE
RESERVED OR NON-RESERVED? R

====> ENTER DATASET NAME: -123456 MYDATA

====> ENTER NUMBER OF TRACKS: 10

====> ENTER SECURITY CLASSIFICATION: ?
Security classification is required. Enter UN for
unclassified, IU for IBM Internal Use Only, IC for IBM
Confidential, or IR for IBM Confidential Restricted.
If you want to classify a file as IC or higher, the
file must be reserved (negative account number).

====> ENTER SECURITY CLASSIFICATION: IU

YOUR DATASET HAS BEEN CREATED WITH THESE CHARACTERISTICS:

| STATUS   | PERMANENT |
| DSN      | -123456 MYDATA |
| RECFM    | F |
| BLKSIZE  | 6233 |
| LRECL    | 6233 |
| SPACE    | (19069, (10, 0)) |
| VOLUME   | APL531 |
| UNIT     | 3350 |
| CODE     | A |
| SECURITY | IU |

You may now wish to format the dataset with a particular block size which your application needs; that's easily done by using INITIALIZE in the same workspace:

```
INITIALIZE
DATASET NAME: -123456 MYDATA
BLOCK SIZE: 19069

10 RECORDS INITIALIZED
Classifying Existing Datasets

You have seen how new files will get classified as they are created, but there will always be some files which were not classified at the time of creation on our system—perhaps they are files which were transmitted to us over the network, or created or installed by a batch job. These datasets will not carry any security classification until you manually enter one for them. In no cases do we ever supply a default classification; it’s always up to you to say what the classification is.

On pages 73-78 we discussed workspace `FILELIB`, which is used for seeing what TSIO datasets you have. The same workspace may be used for classifying those datasets. Several new functions have been added to make classification as simple as possible. The new functions include `CLASSIFY`, `SORTBYCLASS`, and `CLASS`.

`CLASSIFY` will allow you to classify your TSIO datasets. It takes as a right argument a matrix of filenames (FILES) or a character string. The character string can either be a single filename or a qualifier for a group of files. You may use the `GROUPS` function to find out what groups you have and classify those all at once using the first level qualifier as the argument to `CLASSIFY`.

- `CLASSIFY FILES` — (all files)
- `CLASSIFY 'ETC'` — (all files beginning with “ETC”)
- `CLASSIFY 'ETC'` — (a single file named “ETC”)

There are two different ways you can classify your files. You can classify them all at once, giving all files the same classification, or you may classify them individually. If you are using a 3270-type terminal, you will automatically be presented with a full-screen panel to do the classifying.

You may give your datasets one of four classifications:

- 0 = Unclassified
- 1 = For IBM Internal Use Only
- 2 = IBM Confidential
- 3 = IBM Confidential Restricted

Note: Registered IBM Confidential is not supported on Kingston APLSV. You cannot assign this classification to any dataset on the system.

The function `SORTBYCLASS` is used in conjunction with `STATUS` and `USAGE`, to give a list of filenames sorted by their classifications. Files not yet classified are listed first. Following this the files are listed from the highest classification (ICR) to the lowest (UNCL).

Some examples of using `SORTBYCLASS` are:

- `SORTBYCLASS STATUS FILES`
- `SORTBYCLASS USAGE 'ETC'`
- `SORTBYCLASS STATUS 'ETC'`
An additional function, \texttt{CLASS} takes as its right argument any of the abbreviations for security class: \texttt{UNCL}, \texttt{IUO}, \texttt{IC}, and \texttt{ICR}. \texttt{CLASS} will return a listing of only those files with that specific classification. Some examples of its use are:

\begin{verbatim}
SORTBYCLASS STATUS FILES CLASS 'UNCL'
CLASSIFY 'ETC' CLASS 'IUO'
SORTBYCLASS USAGE FILES CLASS 'IC'
\end{verbatim}

If you have any problems or questions regarding these functions, please feel free to call the APL User Support Group at any time for assistance. The APL Hotline number is T/L 695-1234.

You can also use these functions to review the security classifications already given to your files. You should periodically make sure that the original classification of the file is still accurate.

Corporate Security Instruction 104A requires that a self-assessment be conducted \textit{at least} annually, and further, that you, the user, be able to demonstrate that you have done so. A means of meeting this requirement would be to keep a dated listing of your file names and their classifications in your desk so that you can produce it if you or your management are audited.

We have tried to make the dataset classification procedures simple, straightforward and fast. We have provided the tools, but it is your responsibility to do the actual classifications and to be in compliance with Corporate Security requirements.

As always, if you have any questions or comments regarding these facilities, feel free to call us at T/L 695-1234.
Deleting TSIO Datasets

If you wish to delete any file, active or archived, it's easy to do that. First of all, we should point out that the active files are costing you a penny per track per week to store them, while the archived files are free... there is no storage cost whatsoever for having your files held here in archives. Therefore, if you even suspect that you might ever want to get the files back, we recommend that you leave them in archives... just be sure to classify them; that only takes a minute or so to do. (For more information on dataset archiving, type "ARCHIVEHOW" in workspace "1 FILELIB").

However, if you are very sure that you never will want the archived data again, you can easily delete it by using the DELETE function which we provide for this purpose: "LOAD 1 FILELIB" and type "DELETE". You will be prompted for the file name. This will now delete any file (active or archived).

In particular, please realize that you do not need to have an archived file brought back on-line in order to delete it.

*It all sounds simple, but BEWARE....*

Be aware of a special requirement here: If you had been storing IBM Confidential Restricted data in the file which you are now planning to delete, Corporate Security Instruction 104A stipulates that the data "must be erased or overwritten when it is no longer required" [See "Disposal of Unencrypted Residual Information", on page 143]. There is a possible problem which this procedure will prevent. It is very unlikely-but technically possible— to have a user allocate a new dataset and find that it has another user's data already in it.

Realize further that this problem has nothing to do with APL: the same requirements hold for any other language running under the MVS Operating System. The situation could occur if a user deletes a dataset and the next user allocates a dataset which happens to be put on the space on the disk which was just freed up, and creates a dataset which has exactly the same blocksize, the same record format, and the same logical record length. If that situation occurs, and if the new user reads the dataset before he writes any of his own data to it, it is technically possible to read the data which belonged to the old dataset which was supposedly deleted.

Realize that “deleting” a dataset doesn’t actually remove any data from the disk pack... it simply sets a flag which says that that space is now available for re-use. To be absolutely sure that no one else can retrieve your data, the data should be overwritten before you attempt to delete it.

This can be done with code which you supply for yourself, or it may be handled by using the function called "INITIALIZE" in workspace 1 AIDS. This is the same function which is used to format a dataset which you have just created with the ALLOCATE function. The INITIALIZE function simply writes the block number to each block, which overwrites (and destroys) the previous data.
)LOAD 1 AIDS
SAVED 10.22.11 08/24/84

INITIALIZE
DATASET NAME: -123456 MEANING.OF.LIFE
BLOCK SIZE: 19069

100 RECORDS INITIALIZED.

DELETE
ENTER ACCOUNT NUMBER (NEGATIVE FOR RESERVED DATASETS) AND DATASET NAME:
DELETE DSN=-123456 MEANING.OF.LIFE
ON-LINE DATASET '1'123456 MEANING.OF.LIFE' DELETED
Dataset Privacy: Using Command Datasets

Reserved versus Non-Reserved Datasets

Each of your TSIO datasets has your APL account number associated with it. While it is possible to read and write your own datasets without specifying an account number, as in "CTL+'SR DSN+DATA'", a more complete form of the same command for a user with APL account number 123456 would be "CTL+'SR DSN+123456 DATA'".

While the system does allow you to leave off the account number, it's always a good practice to include your account number with the dataset name when you write any code, so that it can be executed by another user. In that way, multiple users could use your data. Since any other users are given access to your data, this is referred to as a non-reserved dataset. For example, a non-reserved dataset name is "123456 MYDATA". Any non-reserved dataset can be accessed freely by anyone on the system. Only non-confidential data should be stored in this type of dataset.

A reserved dataset is a dataset that can be accessed only by its owner... it is said to be "reserved" for your own usage. A reserved dataset name looks like this: "~123456 MYDATA". If you are signed on with any account number other than the account number that the reserved dataset is under, an attempt to access it will result in a return code of "2 - RESTRICTED COMMAND".

So far, we have established that a reserved dataset is distinguished from a standard (non-reserved) dataset by means of the account number: for a non-reserved dataset, it's a positive number that matches the account number; for a reserved dataset, it's the negative version of that same account number. If you don't specify an account number at all, it defaults to a non-reserved dataset (anyone can access it, if they know the name).

And yes, you can have one of each of those datasets with the "same name"... that is, you could have both "123456 MYDATA" and "~123456 MYDATA". Since the account number is part of the dataset name, those are completely different dataset names, and will comfortably co-exist.

Well, a reserved dataset seems to be a nice security feature, but there's one major problem: if it keeps out everyone except the owner, that's somewhat overly restrictive. All of the processing for any sensitive project would have to take place on a single account number. So — of course — there's an extension to that.

Command Datasets

A command dataset is a special instance of a reserved dataset; that is, a command dataset is always reserved, but further, it's a special dataset which contains (you guessed it) commands and lists of user numbers that can use each of the commands. [See the function called "IC" in workspace 1 AIDS for building and maintaining command datasets.] You can put any TSIO command that you wish in your command dataset, and may specify exactly
who may use each of the commands. When they use them, they are executing them just as if they were under your account number; that is, they have your access to those commands, and so can access your reserved datasets just as if they were you... if you authorize them.

If you are user 123456, you’d get this response:

\[
\text{CTL}+'\text{SR DSN=}'123456\ \text{MYDATA}' \\
\text{CTL} \quad 0 \quad \text{----- (successful)}
\]

But if you are any other user, you’d get this:

\[
\text{CTL}+'\text{SR DSN=}'123456\ \text{MYDATA}' \\
\text{CTL} \quad 2 \quad 1 \quad \text{----- (restricted command)}
\]

No problem. If you want to let a certain user (or even all users) have a specific type of access to your dataset, you can put that same command into a command dataset, and authorize the appropriate users. Then, they would access it like this:

\[
\text{CTL}+'\text{IC DSN=}'123456\ \text{GATE(7)}' \\
\text{CTL} \quad 0 \quad \text{----- (successful)}
\]

In this command, the name of the command dataset is "123456 GATE", and the command that this user is authorized to access has been put into command slot 7 of that dataset. The command that’s being used is IC... “Indirect Command”.

So far so good. What more could you ask for? Well, just one more problem: the command that appears in the command dataset had to be a complete command, not just a portion of the command. Therefore, if you wanted to let certain people access any of your twenty datasets for indexed-read only, you had to have twenty commands. Hmmm, seems like there ought to be a better way... and so, of course, there is:

**Symbolic Parameters in Indirect Commands**

A *symbolic parameter* can be thought of as being, in many respects, analogous to an APL variable. What’s often needed is a way to put most of the command in the dataset, and stipulate what parameters may be added, but then allow the user to supply values for those parameters.

A symbolic parameter is distinguished from other TSIO terms and values in that it begins with "II" (the APL version of what JCL sees as an ampersand: "&"). This term would then be used both in the command within the command dataset and in the command that calls that one. For example, assume the command in the command dataset to be "SR DSN=123456 ABC,DISP=\&D". When this command is invoked, the command would look like this:

\[
\text{CTL}+'\text{IC DSN=}'123456\ \text{GATE(7)},\&D=\text{SHR}' \\
\text{CTL} \quad 0
\]
Notice that the “\texttt{DISP=AD}” in the command within the command dataset will be filled in with the value supplied by the user, “\texttt{\^D=SHR}”, allowing TSIO to read the string as “\texttt{DISP=SHR}”.

If you are a user who is entering the \texttt{IC} command, the name of the symbolic parameter that you enter must, of course, match the name that appears in the command dataset. That name can be from one to eight characters long, alpha-numeric (A-Z and 0-9). The first character may not be numeric. The first three characters may \textit{not} be \texttt{SYS... that's a reserved prefix, which gives us some additional features.}

\textbf{Reserved Names for Symbolic Parameters}

Several special names have been established which the system will replace with a value upon their use (a table of these terms follows shortly). For example, one such term is \texttt{\^SYS\_DATE}. This term returns the current date in the form “YYDDD” (Julian date). If you wish to be able to create a new dataset whose name contains the current date, this can be done by placing a command in a command dataset like this:

\begin{verbatim}
SW DSN=123456 \^SYS\_DATE
\end{verbatim}

If this were the 250th day of 1984, the system would then treat the command as though it were

\begin{verbatim}
SW DSN=123456 D84250
\end{verbatim}

Notice that we put an alphabetic character (“\_”) in the command so that the dataset name won’t start with a numeric. The symbolic name may be used to replace any \textit{portion} of a term (but it \textit{can’t} be used to pass more than one term). For example, consider the following commands:

\begin{enumerate}
\item \textbf{Example 1:}
  \begin{itemize}
  \item Command dataset contains this: \texttt{IR DSN=123456 \^SYS\_DATE}
  \item User enters the command as this: \texttt{IC DSN=123456 GATE(7)}
  \item and TSIO sees command as this: \texttt{IR DSN=123456 D80250}
  \end{itemize}
\item \textbf{Example 2:}
  \begin{itemize}
  \item Command dataset contains this: \texttt{IR X DSN=123456 MINE}
  \item User enters the command as this: \texttt{IC DSN=123456 GATE(8),\^X=RW}
  \item and TSIO sees command as this: \texttt{IRW DSN=123456 MINE}
  \end{itemize}
\item \textbf{Example 3:}
  \begin{itemize}
  \item Command dataset contains this: \texttt{SR DSN=123456 WEEK\_WK}
  \item User enters the command as this: \texttt{IC DSN=123456 GATE(9),\^WK=19.B}
  \item and TSIO sees command as this: \texttt{SR DSN=123456 WEEK19.B}
  \end{itemize}
\item \textbf{Example 4:}
  \begin{itemize}
  \item Command dataset contains this: \texttt{SR \textit{\texttt{DISP=AD}},DSN=123456 ABC}
  \item User enters the command as this: \texttt{IC DSN=123456 GATE(12)}
  \item and TSIO sees command as this: \texttt{SR DSN=DSN=123456 ABC}
  \end{itemize}
\end{enumerate}
Trivia Department: Notice that a specification arrow ("\(\rightarrow\)") can always be used in place of an equal-sign in any TSIO command... TSIO will make the substitution.

Notice also that the command dataset won't see any value from a symbolic name if that name isn't specified; that's okay, the system will supply the default value. In the example used here, TSIO would treat the command as though it had been entered as "SR DISP=SHR,DSN=123456 ABC".

If you would prefer to specify your own defaults, you can do that: just enter the command in the IC dataset like this: BLKSIZE+ABLK, ABLK+19069.

### Reserved Names for Symbolic Parameters

<table>
<thead>
<tr>
<th>Term</th>
<th>Purpose</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SYSACCT</code></td>
<td>Account number</td>
<td>123456</td>
</tr>
<tr>
<td><code>SYSPACCT</code></td>
<td>Positive account number (same)</td>
<td>123456</td>
</tr>
<tr>
<td><code>SYSNACCT</code></td>
<td>Negative account number</td>
<td>-123456</td>
</tr>
<tr>
<td><code>SYSPCODE</code></td>
<td>“Scramble” of positive account</td>
<td>AAABOCEA</td>
</tr>
<tr>
<td><code>SYSNPCODE</code></td>
<td>“Scramble” of negative account</td>
<td>PPPOBNMA</td>
</tr>
<tr>
<td><code>SYSTIME</code></td>
<td>Time of day (HHMMSS)</td>
<td>152148</td>
</tr>
<tr>
<td><code>SYSHOP</code></td>
<td>Hour (HH)</td>
<td>15</td>
</tr>
<tr>
<td><code>SYSMIN</code></td>
<td>Minute (MM)</td>
<td>21</td>
</tr>
<tr>
<td><code>SYSEC</code></td>
<td>Second (SS)</td>
<td>48</td>
</tr>
<tr>
<td><code>SYSDATE</code></td>
<td>Date (YYDDDD)</td>
<td>84245</td>
</tr>
<tr>
<td><code>SYSYEAR</code></td>
<td>Year (YY)</td>
<td>84</td>
</tr>
<tr>
<td><code>SYSJDAY</code></td>
<td>Day of year [Julian day] (DDD)</td>
<td>245</td>
</tr>
</tbody>
</table>

Note: "SYS" is a reserved prefix; no user-generated symbolic names may begin with "SYS". Also, no specification is allowed to these names.

These names (and all symbolic parameters) may only be used by a command within a command dataset; they may not be invoked directly.
The “scramble” of the account number is an encoding that TSIO uses to name datasets. Although you see your dataset as “123456 MYDATA”, TSIO sees it as “TSIO.AAABOCEA.MYDATA”. (While you (as a standard “Level 10” user) can’t specify this name directly through TSIO, knowledge of its existence may be helpful for dealing with datasets moving on and off the system.) All TSIO datasets start with “TSIO.”, to identify them, and the next eight characters are an encoding (or “scramble”) of the account number: 123456→AAABOCEA, and -123456→PPPOBNMA. The functions for performing this translation are as follows:

```
v Z+SCRAMBLE N
[1] Z←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'[IO+(8⍪16)×N]
v
SCRAMBLE 123456 123456
   AAABOCEA
   PPPOBNMA
pSCRAMBLE 123456 123456
   2 8

v Z+UNSCRAMBLE N;IO
[1] IO←0
[2] Z+N
[3] (¬∧/2⍴'ABCDEFGHIJKLMNOPQRSTUVWXYZ')/0
[4] (8⍪2)/0
[5] Z+16×'ABCDEFGHIJKLMNOPQRSTUVWXYZ'⍪Z
v
UNSCRAMBLE 'AAABOCEA'
123456
```
Using Symbolic Parameters as Pseudo-Passwords

Let's assume for a moment that you have done all of your homework as far as using IC datasets and setting up symbolic parameters, so that the access that you provide for other people to access your datasets seems pretty secure. Congratulations. But, just how secure is it? Could we be overlooking anything?

APLSV currently provides an environment in which the names of TSIO datasets cannot be acquired by another user, and if you lock the APL functions that are used to access those datasets, the users can't even see the name of the command dataset.

Let's consider that for a moment; that's a pretty important point. If any of the people who use your APL functions could discover the name of the command dataset that you are using to access your sensitive data, they could simply issue the IC command to use the command dataset directly, without going through your functions. And if your functions do anything to reduce the amount of data that each user is authorized to see from that file—for instance, if each user is only authorized to see his own data—all bets are now off. If he can find the name of that command dataset, and if he is in the authorization list for any of the commands, he could sidestep your functions and look at everything.

But how could he find the name of the command dataset? We have already established that APLSV prevents the users from unlocking the functions, and also prevents them from acquiring the names of anyone else's datasets. So what's the problem?

Well, we are not going to be using APLSV forever. We already announced (in July of 1984) that our plans are to migrate to APL2 under TSO. And the TSO environment doesn't protect dataset names. If the entire protection of your data comes down to the supposition that other users don't know the names of your datasets, you may be setting yourself up for problems on future systems. What would be nice to have would be a password on the TSIO dataset, as one extra measure of protection. While there is no password facility available for TSIO datasets per se, you can use symbolic parameters to create an equivalent capability. If you set up the name for the symbolic parameter as something obscure, and then assign it a value which would be necessary to create a complete, valid command, you will have created a facility in which that command cannot be used unless you know how to patch it together. Remember, a user who is authorized to use a command in a command dataset can't see that command, he can only execute it.

Here's how this procedure can be made to work:

Let's assume that the command that you issue against your command dataset looks like this:

\[ IC \ DSN=-123456 \ SALES.ACCESS(12) \]

...and the command that it executes currently looks like this:

\[ SR \ DSN=-123456 \ SALES.FILE \]
What can we do to protect this a little bit further? Well, if the command is changed to this...

```
S\NEWYORK DSN=-123456 SALES.FILE
```

...it would only be executable if the command was issued like this:

```
IC DSN=-123456 SALES.ACCESS(12),\NEWYORK=R
```

If that "NEWYORK" was omitted, the command would become simply "S DSN=-123456 SALES.FILE", and wouldn't be valid.

A user could only supply that parameter if he knew the inner workings of your code and commands. Just knowing the name of the command dataset is no longer enough.

Notice that the real "password" portion of all of this is the name of the symbolic parameter just as much as its value. Here are some other examples of this technique:

---

**Example 1:**

Command dataset contains this: `IR DSN=-123456 SALES.FILE,DISP=SH\DOG`

User enters the command as this: `IC DSN=-123456 GATE(7),\DOG=R`

and TSIO sees command as this: `IR DSN=-123456 SALES.FILE,DISP=SHR`

[If the user doesn’t supply `DOG`, the command will default to `DISP=SH`, and fail.]

**Example 2:**

Command dataset contains this: `IR DSN=-123456 MINE,CODE=\ETC,\ETC=Q`

User enters the command as this: `IC DSN=-123456 GATE(8),\ETC=A`

and TSIO sees command as this: `IR DSN=-123456 MINE,CODE=A`

[If the user doesn’t supply `ETC`, the command will default to `CODE=Q`, and fail.]

---

By using a "pseudo-password" procedure like this, you’ll be ahead of the game, with an extra step of protection to keep the data secure even if the command dataset’s name is compromised.
Creating and Maintaining IC Datasets

Workspace 1 AIDS contains lots of general-purpose functions for assisting you with utilitarian types of tasks which you may need to do, particularly those related to the use of TSIO for creating, initializing, renaming, or deleting datasets.

One of the main functions that is offered in 1 AIDS is the "IC" function. That function provides the ability to display, alter, and maintain TSIO command data sets. The function will guide you via menu selection and prompting for the various services you might require.

The following functions are available within the IC facility to manage command data sets:

---

**Commands for using the IC facility:**

- SET, DISPLAY, and DROP .................. manage TSIO commands
- ADD, COPY, GIVE, REPLACE,
  DELETE, and PURGE ...................... manage user account numbers
- ACCESS, WHO, NO, LIST, and USERS. provide information about the command dataset
- HELP, END and QUIT ..................... miscellaneous functions to help you to use the IC facility

---

"By indirections find directions out."

—Hamlet II, i, 63
William Shakespeare [1564-1616]
Here’s the action of each of those commands:

- **SET index-number**
  ...will prompt for a TSIO command and index number, if not supplied. That command will be put into the command data set at the selected index number, replacing any previous command at that index number.

- **DISPLAY index-number(s)**
  ...will display the selected TSIO commands.

- **DROP index-number(s)**
  ...will delete the selected TSIO commands. The command will be re-initialized and user authorization to it withdrawn.

- **ADD**
  ...will add one (or more) user(s) to one (or more) TSIO command(s). Previous users are unaltered. You will be prompted for command index numbers and user account numbers.

- **COPY**
  ...will add users of one TSIO command to another. You will be prompted for a FROM command index number and TO command index number(s). Previous user account numbers on the TO command(s) will be unaltered.

- **REPLACE**
  ...will replace users from one TSIO command to another. You will be prompted for a FROM command index number and TO command index number(s). Previous user account numbers on the TO command(s) will be deleted first.

- **GIVE**
  ...gives a user the same access that another user already has.

- **DELETE**
  ...will delete one (or more) user(s) from specified TSIO command(s).

- **PURGE user-account-number(s)**
  ...removes specified user account number(s) from the authorization matrix; therefore, it effectively removes authorization from all TSIO commands and removes any trace of the user from the command data set. Since only users not listed in the authorization matrix are represented by user number 0, PURGED users will then be able to use commands with a user number 0.

  **Note:** It is redundant to precede **PURGE** with **DELETE**.

- **WHO index-number(s)**
  ...will display the user account numbers authorized to use specified TSIO commands.

- **NO index-number(s)**
  ...will display the specified TSIO command(s) and its authorized users.
• **LIST option**
  ...will list all TSIO commands in the data set and the user authorized to use each. You will be prompted for one of three options, if not supplied:

  - **INUSE** will list only those entries containing commands or users
  - **EMPTY** will list entries containing neither commands nor users
  - **ALL** will list all entries

• **USERS**
  ...will list all user account numbers referenced in this data set.

• **HELP command**
  ...describes a selected command. You may type **HELP** followed by a command (for instance, **HELP DROP**), or just **HELP** and you will be prompted for the command that you wish to have described. A question mark, “?”, may also be typed in response to any prompt, and additional instructions will be provided.

• **ACCESS user number**
  ...will prompt for the user number if not supplied. A list of the number of the commands that the user is authorized to access will be supplied.

• **END (or a response of ENTER-key only)**
  ...will save your changes on the file and exit from the IC function.

• **QUIT (or a right arrow, “>”, entered at any prompt)**
  ...exits from the IC facility without modifying the file.

In response to any prompt:

• A question mark, “?”, will provide more information.
• An empty response will normally return to the primary menu with no further action.
• A right arrow, “>”, will interrupt the function.

For any function name or option, you may specify only as much of the word as is needed to uniquely identify it. For example: “HE US” is the same as “HELP USERS”, but “D” is not enough (since it could mean **DELETE**, **DISPLAY**, or **DROP**).

**New IC Datasets**

If the IC dataset you try to use does not exist, you will be given the option of having it created. New IC datasets are always created as permanent datasets, using the **ALLOCATE** function (described on page 79). If you elect to have a new data created, you will be prompted for the number of users, the number of commands you expect to put into the file, and the security classification of the file. The numbers you give will be adjusted upward to completely fill the number of tracks your specifications require.
Command Datasets — The Inner Workings

This discussion of command datasets is included for the benefit of the APL programmers who wish to understand the inner workings of the facility. It is not recommended reading if you are simply a user of the facility, and not a programmer.

 Provision for enhanced security of datasets created under TSIO is embodied in the use of reserved datasets with controlled access. A dataset is said to be reserved to a user when the name of the dataset is qualified by the negative of the user's account number. This user can access the dataset freely, but other users can access it on a restricted basis only, subject to an explicit authorization table, as follows:

 A command dataset is a reserved dataset, but one which is available to other users solely for the purpose of letting a command be executed indirectly on behalf of those other users. The other users have access to the command dataset, and certain commands in them, only through being enrolled in an authorization matrix in the command dataset, as described later on.

 A dataset reserved to a particular user can be accessed by other users for reading and writing only indirectly, through the mediation of a command dataset reserved to that user.

 Consider the case in which user 123456 wishes to authorize user 246810 for indexed reading of the reserved dataset 123456 SECRETS. To accomplish this, user 123456 stores an image of the command IR DSN=-123456 SECRETS as, say, the eleventh member of the reserved dataset -123456 GATEKEEP. Subsequently, after sharing a control variable with TSIO in the normal way, user 246810 may submit a request for an indirect operation, using an IC command:

         CTL='IC DSN=-123456 GATEKEEP (11)'

 On receipt of this command, TSIO will first determine that -123456 GATEKEEP has the proper format for a command dataset, then check the authorization matrices within it to determine whether user 246810 is permitted to utilize command form 11. Only if both of these checks are satisfactory will TSIO execute the read command on behalf of user 246810, who may then proceed with normal reading of -123456 SECRETS.

 If the dataset -123456 GATEKEEP is not properly constructed as a command dataset, or if user 246810 has not been authorized to use the eleventh command, then TSIO issues the return code 2 1 - RESTRICTED COMMAND to user 246810.

 A command dataset must have CODE=A, RECFM=F, BLKSIZE=320.
Record 0 of the command dataset must be an integer vector with the following elements:

- [0] First authorization matrix record number
- [1] Last authorization matrix record number
- [2] Maximum number of rows in authorization matrices
- [3] Last block allocated for authorization matrices
- [4] Number of authorized user entries, total for all authorization matrices
- [5] Creator’s user number
- [6 7 8 9 10 11] Year, month, day of month, hour, minute, second of last update of the dataset

Referring to that initial record as "RECO", records 1 through "RECO[0]-1" of the dataset provide for indirect commands. From record "RECO[0]" on the records are authorization matrices of "32+RECO[0]-1" columns and of a number of rows no greater than "RECO[2]".

Columns 0 through 31 of the matrices contain the (32-bit) base two representation of user identification numbers. Columns 32 and on correspond to records 1 through "RECO[0]-1" of the dataset: if the element in row I, column 32+J is 1, the command in record J may be executed on behalf of the user identified in columns 0 through 31 of row I.

An entry for user 0 represents a general authorization for users other than those listed in the authorization matrices. A 1 in column 32+J of this row permits such users to execute the J-th command.

Each authorization matrix except the last must have the maximum number of rows as given by "RECO[2]". The rows of all the authorization matrices, considered as one matrix, must be in order by the base-two values of columns 0 through 31. The number of columns in the matrix must be a multiple of 8.

Command datasets are created and maintained by means of the function called "IC" in the workspace 1 AIDS on the Kingston APLSV system. The necessary instructions for their use are detailed on pages 92-94, and are also found under ICHOW in that workspace.
Semaphores: A General Enqueue/Dequeue Facility

When discussions of "security" come up, most users wince and consider that it means that they'll have to do something else to comply with Corporate Instruction 104A. Well, "security" can also imply data integrity; that is, ensuring that a user can't accidentally lose or damage data. One place where potential problems becomes very visible is in the simultaneous updating of a single file by many users. What's needed is a means of guaranteeing that a given operation by one user runs to completion before another user is allowed to start a new operation. In data processing circles, this is called an enqueue/dequeue facility.

"Semaphores" provide this general enqueue/dequeue facility for TSIO. They enable APL users, through the TSIO file processor, to notify each other that they wish to reserve certain resources. The users of a common resource agree on an integer value that will represent that resource and, before using the resource, check the enqueuing status of that value to see if they may have control of it.

When two or more users can modify the same file, problems arise if they attempt to modify the same record simultaneously. Assume that user 1234 has read a particular record, with the intention of modifying it. While he is modifying it, user 5678 reads the same record with the same intent. User 1234 writes his new version back to the file, but then user 5678 writes his new version also. What happens is that the changes made by user 1234 are lost, because his version is overwritten by user 5678's version.

What is needed is some programmable way for users to communicate with each other, so that all users but one may be inhibited from updating some record or group of records. Semaphores provide that means. During TSIO indexed operations (IR or IRW), they are controlled by the TSIO operation codes 2 3 4 and 5; for example, CTL+3 10.

There are lots of applications in which the same dataset must be used asynchronously by a number of different users. To a certain extent this can be managed without conflict by appropriate use of the disposition parameter (such as "DISP=SHR" or "DISP=OLD"), but this alone may not be completely satisfactory. On the one hand, if the dataset is opened for exclusive use by one user (DISP=OLD), no matter how small the portion of the records actually involved, no other user can access any part of it until the dataset has been closed and reopened. On the other hand, if the dataset has been opened for shared use (DISP=SHR), except for the simplest case, in which all the sharers are only reading, some further mechanism is necessary to provide more dynamic control and finer resolution of the shared use.

Such a mechanism is provided by the operation codes 2 3 4 and 5, which may be used on the control vector (CTL) when a dataset has been opened for indexed access with DISP=SHR. When one of these operation codes is used, the second element of the control vector may have any non-negative integer value. Except for zero, the significance of this value is completely arbitrary, although
commonly it may be a record number or a coded identification of a set of records.

<table>
<thead>
<tr>
<th>Code</th>
<th>Key</th>
<th>Use</th>
<th>Effect</th>
<th>Action if in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>n</td>
<td>Exclusive</td>
<td>Immediate</td>
<td>Return code 15</td>
</tr>
<tr>
<td>3</td>
<td>n</td>
<td>Exclusive</td>
<td>When Available</td>
<td>Delay until available</td>
</tr>
<tr>
<td>4</td>
<td>n</td>
<td>Shared</td>
<td>Immediate</td>
<td>Return code 15</td>
</tr>
<tr>
<td>5</td>
<td>n</td>
<td>Shared</td>
<td>When Available</td>
<td>Delay until available</td>
</tr>
</tbody>
</table>

Note: "n" may be any integer from 1 through $$1+2*31$$. The semaphore is released by assigning a key of 0 with any of the semaphore codes, or by assigning a new semaphore, or by breaking the sharing [through retraction, ")LOAD", ")CLEAR" or ")OFF"].

Operation codes 2 and 3 signify that the user wishes to have exclusive use of the facility denoted by the second element of the control vector, either immediately (2) or whenever it becomes available (3). The operation codes 4 and 5 signify that the user wishes to have non-exclusive use of the facility, — on a shared basis, simultaneously with other users — either immediately (4) or whenever it becomes available for such use (5). In any case, once the use of a facility has been acquired, the user is said to have an exclusive hold or a shared hold on that facility.

If more than one control variable has been attached to a dataset, each one may independently have a hold on a facility associated with that dataset. An existing hold associated with a control variable is automatically cancelled by any subsequent use of one of the operation codes 2 3 4 5, whether or not the new request is fulfilled. An existing hold can also be cancelled explicitly by using a zero for the second element of the control vector with any of these operation codes.
Consider for example the following functions, several instances of which are assumed contending to update records 100-120 of a dataset called “EXAMPLE”:

\[
\begin{align*}
\text{UPDATE} ; I \\
\# \text{ UPDATES RECORDS 100-120 OF THE 'EXAMPLE' DATASET} \\
\text{TOCTL 'IRW DSN=1234 EXAMPLE'} \\
\text{TOCTL 3 7} \\
I+0 \\
L1:\text{120}-(99+I+1)/L2 \\
\text{DAT+RECORD I} \\
\text{TOCTL 1 ,I} \\
\rightarrow L1 \\
L2: \text{TOCTL 3 0} \\
\end{align*}
\]

\[
\begin{align*}
\text{TOCTL X ;E} \\
\# \text{ PASSES CONTROL INFO TO SHARED VAR; READS RETURN} \\
\text{CTL+X} \\
(0 - E - CTL)/0 \\
' TSIO RESPONSE : ' , \text{E} \\
. \\
\end{align*}
\]

The updating is performed by the function \text{RECORD}. The contender that achieves exclusive hold on facility 7 of the \text{EXAMPLE} dataset will next update records 100-120.

If a request for an immediate hold on a facility cannot be fulfilled because the facility is being held by another user, the control variable is returned with the value 15. Note that a request for exclusive use may be denied because another user has either an exclusive hold or a shared hold, and that a request for a shared hold may be denied only if another user has an existing exclusive hold. A request for a hold with codes 3 or 5, indicating that the user can wait, will remain in force until the facility becomes available for the type of hold requested, unless cancelled by a strong interrupt. The fulfillment of the request is signified by the value 0 in the control variable, which is, as usual, interlocked against reading until TSIO respecifies its value following receipt of the request.

Two points are worth noting here: First, the use of these operation codes \textit{does not automatically guarantee protection} against conflicting use or other problems associated with shared facilities. It merely provides the potential for such protection to be incorporated in an application program. Thus, for example, there is no built-in constraint against the use of the 0 and 1 operation codes for reading or writing into a record which is otherwise part of a facility held by another user.

Second, because of the absence of such built-in constraints, the definition of the facilities denoted by the second element of the control vector when using an operation code of 2 3 4 or 5 is completely up to the application program, and need not refer in any way to the dataset in use. In such a case the dataset acts as a communication device with certain useful properties. The application will be making use of the built-in queuing facilities.
of TSIO, as well as the properties of the operation codes, to control the use of some common resource.

**Shared Semaphores**

Kingston APL consists of several separate systems, but don't worry: semaphores are shared between all of the systems.

Because of this, you don't have to be concerned about updates to your datasets occurring simultaneously on multiple machines. The simple answer is, “It handles it”. Our goal with Kingston APL is always to present a “single-system image” [good IBM buzz-words] to our users. All workspaces, datasets, and semaphores are shared between all of the systems, so that it shouldn't matter which system you are signed on to.

*Trivia Department:* We might mention that, even though the APL designers have always conversationally referred to this facility as “semaphores”, none of the APL manuals have ever called them that... in fact, they haven't called them *anything*. Admittedly, this makes it rather hard to find them in the index [...but you'll find them in ours.]

---

100  
Section 6: Protecting Your TSIO Datasets
Who Is Using Your TSIO Datasets?

Workspace 1 SMF retrieves data showing a summary of accesses to each of your TSIO datasets during the past week (from early Saturday morning until early the following Saturday morning). The data is summarized by file name and accessing user-ID.

First enter SMF to retrieve the data, then enter SHOW ALL to display it.

Data displayed will be:

- the *account number* that accessed a given dataset
- the number of times that he *accessed* that file
- the number of *reads* (or writes) that he issued
- the number of *bytes* of data that he transferred
- the name of the dataset that he accessed (*DSN*)

Additionally, the “SHOW” function may be used in any of these forms:

SHOW DSN 'ABC'
shows accesses to all files whose names start with the string “ABC”. Data is sorted by account-numbers. (“DSN” returns unformatted data, described later.)

SHOW USER 123456 234567
shows accesses by user accounts 123456 and 234567.

SHOW ACCESS SORTED DSN 'ABC'
SHOW READS SORTED DSN 'ABC'
SHOW BYTES SORTED DSN 'ABC'
  Displays the data sorted by the indicated field (largest values first).

Other functions available are:

FILES
returns the names of each of your files that were accessed during the past week (this may not be all of your files).

USERS
returns a sorted list of each of the account numbers which accessed any of your files during the past week (use “SHOW USER n” to determine which files a particular account accessed).
Other Uses for This Information

Many applications have requirements for logging access to the data. Having this information about who is using each of your datasets may alleviate the requirement for writing your own code to do that logging. If your requirement is to simply capture the account number and number of times that the data is accessed, your own audit files may not be needed at all. This already does that.

Also in your favor is the fact that this facility is built into the system; therefore, it cannot be circumvented by some knowledgeable user, and you have the benefit of the logging being done by fast-running system code. Not bad, for free.

Finally, consider the uses of this data even beyond the obvious security uses. For example, let's say that you have slaved away in front of your terminal for months putting together an application that you think has been well-received by your users, but your management may have different ideas. Perhaps it would be advantageous to show them that your application really is being used (...if it is). This workspace may then become an important part of your business case. We use it for that all the time here in Kingston APL Support. To see how well-received a facility really is, there's nothing like gathering up real usage data.
Let's try it out:

)LOAD 1 SMF
SAVED   9.17.44 09/05/84

The first thing that we need to do is to extract the data from the history files, and bring it into the workspace, so that we can look at it. That's done like this:

SMF
***FILE DATE IS 01/26/85 THRU 02/02/85
***TO DISPLAY ALL THE DATA TYPE "SHOW ALL"

If we want to see the usage of all of our files, that's easy; just enter "SHOW ALL" (like it said). But let's assume that we have a lot of files. We don't necessarily want to wade through the usage statistics for all of them. This will show us the usage of each of our files whose names begin with "SALES":

```
SHOW DSN 'SALES'
```

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>ACC</th>
<th>READS</th>
<th>BYTES</th>
<th>DSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>13579</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>123456.SALES.ACCESS</td>
</tr>
<tr>
<td>234567</td>
<td>12</td>
<td>36</td>
<td>11,520</td>
<td>..</td>
</tr>
<tr>
<td>246810</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>..</td>
</tr>
<tr>
<td>666666</td>
<td>3</td>
<td>9</td>
<td>2,880</td>
<td>..</td>
</tr>
<tr>
<td>1123456</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>..</td>
</tr>
<tr>
<td>13579</td>
<td>2</td>
<td>43</td>
<td>182,879</td>
<td>123456.SALES.FILE</td>
</tr>
<tr>
<td>234567</td>
<td>12</td>
<td>33</td>
<td>140,349</td>
<td>..</td>
</tr>
<tr>
<td>246810</td>
<td>2</td>
<td>8</td>
<td>34,024</td>
<td>..</td>
</tr>
<tr>
<td>1123456</td>
<td>2</td>
<td>8</td>
<td>34,024</td>
<td>..</td>
</tr>
</tbody>
</table>

Here is essentially the same inquiry, but with the results sorted by the number of accesses that each user made (and with totals):

```
TOTAL SHOW ACCESS SORTED DSN 'SALES'
```

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>ACC</th>
<th>READS</th>
<th>BYTES</th>
<th>DSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>234567</td>
<td>12</td>
<td>36</td>
<td>11,520</td>
<td>123456.SALES.ACCESS</td>
</tr>
<tr>
<td>234567</td>
<td>12</td>
<td>33</td>
<td>140,349</td>
<td>123456.SALES.FILE</td>
</tr>
<tr>
<td>666666</td>
<td>3</td>
<td>9</td>
<td>2,880</td>
<td>123456.SALES.ACCESS</td>
</tr>
<tr>
<td>13579</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>..</td>
</tr>
<tr>
<td>246810</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>..</td>
</tr>
<tr>
<td>1123456</td>
<td>2</td>
<td>6</td>
<td>1,920</td>
<td>..</td>
</tr>
<tr>
<td>13579</td>
<td>2</td>
<td>43</td>
<td>182,879</td>
<td>123456.SALES.FILE</td>
</tr>
<tr>
<td>246810</td>
<td>2</td>
<td>8</td>
<td>34,024</td>
<td>..</td>
</tr>
<tr>
<td>1123456</td>
<td>2</td>
<td>8</td>
<td>34,024</td>
<td>..</td>
</tr>
<tr>
<td>TOTAL:</td>
<td>39</td>
<td>155</td>
<td>411,436</td>
<td></td>
</tr>
</tbody>
</table>

But hold on a second; there's a suspicious entry here: user 666666 shows up as using the dataset called "SALES.ACCESS", but he doesn't show up as using the dataset called "SALES.FILE". Now,
there's nothing inherently odd about that, but let's suppose that our knowledge of the application points out that the "SALES.ACCESS" file is a command dataset (as described on pages 85-86), and further, let's assume that we know that there happens to be only one command in that dataset: a command which opens the "SALES.FILE" dataset. How is it that this user doesn't show up as using the "SALES.FILE" dataset and the other people do?

If user 666666 is not authorized to access the "SALES.FILE" dataset—that is, if he does not appear in the authorization lists within the "SALES.ACCESS" command dataset—any attempt on his part to use the facility will log the attempt against the command dataset. Here, the command dataset shows usage, because the system can only know whether or not he is on the authorized access lists by actually opening the command dataset and looking at those authorizations. He tried to access it three times, so those three attempts were logged. However, since he is not authorized to use that facility, there is no activity logged against the actual data file ("SALES.FILE"), because he was not able to open that file.

If you really want to see all of the unsuccessful attempts to access your files, that's easy; there's a "FAILURES" function provided for that purpose:

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>DATE</th>
<th>TIME</th>
<th>DSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>246810</td>
<td>01/27/85</td>
<td>23:33</td>
<td>BLIVIT.ACTUAL</td>
</tr>
<tr>
<td>246810</td>
<td>01/27/85</td>
<td>23:33</td>
<td>BLIVIT.ACTUALS</td>
</tr>
<tr>
<td>246810</td>
<td>01/27/85</td>
<td>23:34</td>
<td>BLIVIT.ACTUALS</td>
</tr>
<tr>
<td>246810</td>
<td>01/27/85</td>
<td>23:35</td>
<td>BLIVIT.ACTUALS</td>
</tr>
<tr>
<td>666666</td>
<td>01/30/85</td>
<td>08:33</td>
<td>SALES.FILE</td>
</tr>
<tr>
<td>666666</td>
<td>01/30/85</td>
<td>08:34</td>
<td>SALES.FILE</td>
</tr>
<tr>
<td>666666</td>
<td>02/01/85</td>
<td>07:57</td>
<td>SALES.FILE</td>
</tr>
</tbody>
</table>

Notice that user 666666 tried to access that sales file several times. But more suspicious-looking than that is user 246810... he wasn't even using a command dataset for his access attempts, so he couldn't have accessed the "BLIVIT.ACTUALS" file regardless of what he tried, but it's odd that he was trying this at 11:30 pm on a Sunday night. Perhaps you should ask him about this...

This information also makes it apparent that he misspelled the file name on his first attempt. There is no file under this account called "FILE.ACTUAL"; it's "FILE.ACTUALS" (with an "S") that he was after. His first attempt would have resulted in a TSIO return code of "12 - DATASET NOT FOUND", while the rest of his attempts would have resulted in "2 - RESTRICTED COMMAND". Realize, therefore, that you can trace more than just the attempted accesses to files that you own, you can also see the attempts to guess at names of files under your number even if they don't exist.
Here's another inquiry; it's similar to the first one, but this time we'll sort it by the number of bytes of data that each user transferred while using our files:

```
TOTAL SHOW Bytes Sorted DSN 'BLIVIT'

ACCOUNT ACC READS BYTES DSN

54321 26 26 338,780 '123456.BLIVIT.LOCATION
123456 26 26 338,780
123456 25 25 325,750 '123456.BLIVIT.USAGE
123456 22 22 286,660 '123456.BLIVIT.PARTS
123456 12 12 228,636 '123456.BLIVIT.SURVEY
123456 14 14 182,420 '123456.BLIVIT.CONTROL
123456 12 12 156,360 '123456.BLIVIT.ACTUALS
123456 7 7 91,210 '123456.BLIVIT.INPUT
123456 7 7 91,210 '123456.BLIVIT.PUBLIC
112233 4 4 52,120 '123456.BLIVIT.USAGE
54321 3 3 39,090 '123456.BLIVIT.DESIGN
112233 2 2 26,060 '123456.BLIVIT.LOCATION
54321 2 2 26,060 '123456.BLIVIT.PARTS
112233 1 1 13,030 '123456.BLIVIT.ACTUALS
112233 1 1 13,030 '123456.BLIVIT.CONTROL
54321 1 1 13,030 '123456.BLIVIT.INPUT
112233 1 1 13,030 '123456.BLIVIT.PARTS
112233 1 1 13,030 '123456.BLIVIT.PUBLIC
112233 1 1 13,030 '123456.BLIVIT.DRAWINGS
123456 14
112233 1
123456 14

TOTAL: 198 168 2,261,316
```

(Blank fields up here indicate that the file was opened but that no records were accessed.)
We can easily get a list of all of the users who used any of our files during the past week, like this:

```
USERS
54321 13579 112233 123456 234567 246810 666666 1123456
```

And, of course, we can also look at the inquiry statistics for just a specified user (or for a list of users):

```
SHOW USER 234567
```

```
ACCOUNT ACC READS  BYTES   DSN
234567  6  18    234,540 123456 AARDVARK
234567  6  18    115,200 123456 ACTUALS.FY1984
234567 12  36    11,520  123456 ARTWORK.ACCESS
234567  6  54   509,668  123456 ARTWORK.DESIGN
234567  6  18    17,280  123456 BANANA.ACCESS
234567  6  18         123456 BANANA.KEYWORD
234567  6  18         123456 BANANA.PRICE
234567  5  15     4,800  123456 CENTRAL.ORDERING
234567  6  18     5,760  123456 COST.CONTROL
234567  6  43  135,708  123456 GOOD.YIELDS
234567 14  42  13,440  123456 MAIN.ACCESS
234567 14  619  8,065,570 123456 MAIN.SYSTEM
234567  6  18     5,760  123456 PUBLIC.ACCESS
234567  6  61   794,830  123456 PUBLIC.DATA
234567 12  36    11,520  123456 SALES.ACCESS
234567 12  33  140,349  123456 SALES.FILE
```

For those of you who wish to write your own display functions, or use the data in some other manner, the structure of the data is explained in the workspace; ")LOAD 1 SMF" and type "DESCRIBE".
**Extreme Measures**

Realize that this data is a weekly compilation of the actual data that’s gathered throughout the week. The actual data that’s logged in the system includes such things as the time of day of each access. Since this would result in a potentially huge amount of data to report back to you through workspace 1 SMF, we apply some data reduction first, and end up with what you see here. It’s important for us to point out, though, that there is additional data available here.

Also realize that this report shows the accesses to the datasets which belong to a given account number; this does not show the access that a given user has made to other people’s datasets (the reverse case).

The point to be made here is that the data to satisfy both of these needs is gathered each week. It is not generally available; in particular, it requires a fair amount of programmer activity here to sift through the data for all of the users and manually extract the entries that may apply to a given user or group of users. But at this point, it becomes a management call. If the information is truly needed—not just something of interest—it can be recovered.

Likewise, the only data reported back in this workspace is the information regarding the previous week. That’s all that’s kept on-line, primarily due to the volume of data. But entries from previous weeks are of course saved on tape. If there is a bona-fide need for the data—one which has been cleared through management and Security—data showing times of access, showing access by a given user, and showing activity during previous weeks can be gathered and reported to management.

As always, if you are concerned about potential breaches of security or mis-use of facilities, get us involved.
One method for viewing classified output [albeit not necessarily recommended]
Section 7: Stringent Steps for Stringent Needs

Random Numbers: Understanding Random Link (RL)

Over the years, a source of mystery to occasional users of APL has been the operation of the Random Link. We have had some of our users call us at times, pointing out that there’s a problem with the system, since it appears to be generating the same “random” numbers each time the workspace is loaded. Well, let’s talk about it a bit.

Some people are adamant about the idea that there is really no such thing as a truly random number; everything is traceable back to something. We tend to variously agree or disagree with them, randomly, depending upon our mood....

In APL circles, that “something” that they speak of is the random link. It’s the “seed” from which all of the random numbers grow. You can get right to the seed through RL. In a fresh workspace, RL has a value of 7*5, or 16807. It is used as a part of the calculation to determine what the next “random” number will be whenever the query function (denoted by “?”) is used. And yes, we’ll have to agree that these aren’t really true random numbers... but we’ll show how these “pseudo-random” numbers are even more useful than real random numbers (whatever they are).

---

9 Not quite so large a mystery as the comparison tolerance, perhaps, but an occasional mystery, nonetheless.
The roll function is a monadic function named by analogy with the roll of a die; thus ?6 yields a (pseudo-) random choice from 16, that is, the first six integers beginning with either 0 or 1 according to the value of the index origin (CIO). For example:

```
IO+1
  ?6
  1
  ?6
  5
  ?6
  3
  ?6 6 6 6 6 6 6 6
  4 2 1 5 6 3 4 5 1 1 4 5
IO+0
  ?6 6 6 6 6 6 6 6 6 6 6 6
  0 2 0 2 4 3 5 5 3 0 3 2 4
```

The domain of the roll function is limited to positive integers (selected from $\sim 2 + 2 \times 31$).

The result of the roll function for any number $n$ is approximately $n \times RL \sim 1 + 2 \times 31$. Realize that $RL$ has to be assigned a new value for every random number that’s generated.

But what happens when you re-)LOAD the workspace? Well, since $RL$ has been stored there since your previous use of that workspace, it will have whatever value it had when you )SAVED the workspace. If it’s a workspace that you don’t re-)SAVE (such as a Public Library workspace), the random link always starts at the same point when it’s loaded. And if it hadn’t been changed when the workspace was created, chances are it will quite possibly have the system default value of 16807. That being the case, the sequence of numbers that will be generated for a given series of numbers that you enter will be predictable:

```
)LOAD SAMPLE
SAVED 19.12.47 10/25/84
  1000 1000 1000
  132 756 459
)LOAD SAMPLE
SAVED 19.12.47 10/25/84
  1000 1000 1000
  132 756 459
```

Not too random, is it? But that’s good, because that means that you now have a way to test programs which use random numbers, and duplicate your results.

Since programs don’t remain in their test phase forever (well, some do...), how can you tell it to not do this anymore; in effect, to start generating really random numbers? Aha, that’s what the random link is for.
Notice that we have set ORL to a value which will be different each time that we use it. Here, the seven elements in the time stamp have been added up to produce a single number which is continuously changing.

In the next couple of articles, we will be showing code in which the random link is set to some unpredictable value, such as the summation of today’s date (+/DTS), or the user’s accumulated connect time so far this session (DIIIDQ+2).

In particular, if we set ORL to some particular value which only we know—that almost sounds like a password, doesn’t it?—we can then take a simple numeric encoding of text (such as “OA\T\EXT”) and pass it through the query function, yielding an extremely unpredictable—but repeatable—sequence of numbers. The next two articles will discuss the use of this basic concept to scramble or encrypt text. Since all of the complex number generation is built right in to APL in the form of ORL and “?”, this approach to data encryption can allow very simple APL functions to produce complex (and secure) encryption.

If you are interested in studying random number generation further, just go to any restaurant to observe bistromathics in action (briefly described overleaf). Or, you can refer to The APL Language Manual (GC26-3847) and read about both the roll function that we’ve discussed here, and the dyadic form, the deal function.
Bistromathics

"Bistromathics itself is simply a revolutionary new way of understanding the behavior of numbers. Just as Einstein observed that time was not an absolute but depended upon the observer's movement in space, and that space was not an absolute, but depended on the observer's movements in time, so it is now realized that numbers are not absolute, but depend on the observer's movement in restaurants. ...

And so it was only with the advent of pocket computers that the startling truth became finally apparent, and it was this:

*Numbers written on restaurant bills within the confines of restaurants do not follow the same mathematical laws as numbers written on any other pieces of paper in any other parts of the Universe.*

This single fact took the scientific world by storm...."

—from *Life, the Universe and Everything* © 1982 by Douglas Adams
Used by permission of Crown Publishers, Inc.
Data Scrambling

One step short of real data encryption is data “scrambling”. Scrambling is simply mixing up the order of the characters that you're working with, forming an “anagram” of sorts. Some of you may work with anagrams for relaxation; you know that they can be solved —visually— with no computer equipment. So this technique should not be thought of as being a very secure approach... in particular, this is not a replacement for encryption in places where encryption is required. But for situations in which you may simply want to make the data difficult to read casually, this is a choice that you have. (On the plus side, scrambling the data is much quicker —and therefore cheaper— than encrypting the data.)

This is the “password” for your data. The number can be whatever you wish, from 0 to two less than $2^{31}$ (stated in APL as \$2+2*31\). Whatever value you choose, keep it hidden from your users.

This is called our “mix-up vector”... the number that you choose for this is the range through which the data will be scrambled. In this example, each group of 47 characters will be scrambled. No character will be moved further than 47 characters. Certainly, a larger value is preferable. Of course, since you will need to keep MUV hidden, the users of your application won’t even know its size; that’s part of its protection.

\[\begin{array}{c}
\text{SETUP} \\
\hline
\text{MUV} \\
8 47 25 35 1 31 4 42 30 41 10 14 3 11 23 17 19 40 46 32 29 16 \\
36 38 21 2 9 34 44 45 39 24 12 7 20 5 27 37 15 26 33 43 \\
13 22 28 6 18
\end{array}\]

\[\begin{array}{c}
\text{MUV} \\
5 26 13 7 36 46 34 1 27 11 14 33 43 12 39 22 16 47 17 35 25 44 \\
15 32 3 40 37 45 21 9 6 20 41 28 4 23 38 24 31 18 10 8 42 \\
29 30 19 2
\end{array}\]

Obviously, both the value of \$RL and the MUV variable should be localized in your code, so that users can’t get to them. These two values protect your data. So, here’s the header line of our main function for this application:
These four items have been localized so that we don’t have to worry about little prying fingers tampering with the security measure.

Here are the scrambling and unscrambling functions, named “MIX” (for the function that does the mixing up of the characters, and “XIM” (for the function that does just the reverse of MIX):

Here’s some sample text for us to scramble:

```
TEXT='OPEN THE POD BAY DOORS, PLEASE, HAL.'
MIX TEXT
E PLO,N E PBEO, O SY. RP A DHO E AHL SATD
--- This scrambles the text
NEW=MIX TEXT
E PLO,N E PBEO, O SY. RP A DHO E AHL SATD
--- Realize that a given piece of text will always get scrambled into the same new order, since the value of MUV is fixed.
XIM NEW
OPEN THE POD BAY DOORS, PLEASE, HAL.
--- This unscrambles the text
```
The value that you choose for the length of "MUV" shouldn't be too small, because a very small value will only allow the data to be re-ordered within that small range; for instance, if the value was 2, the only "scrambling" that would be done is that adjacent pairs of characters might be interchanged (or, depending upon the value of R, they might not... — if you call that scrambling!

```
RL+1
MUV+2
```

This is way too small!

Likewise, the value can't be huge, since the length of the output of the MIX function will be a multiple of the length of MUV (causing small chunks of text to be padded to cumbersome lengths), and since MUV itself has to reside in the workspace. A really huge value would be more likely to plague you with WS FULL problems. There are no hard and fast rules here. If you have a use for this technique, the optimal values will become clear through experimentation with your own data.

Additional Reading

If you are interested in the subject of data scrambling and encryption (and who isn’t?), a fascinating book that you should certainly acquire is "Mr. Babbage's Secret: The Tale of a Cypher—and APL," by Ole Immanuel Franksen (published by Strandbergs Forlag, Birkerød, Denmark, 1984).

This entertaining book discusses such topics as the cracking of enciphered advertisements from the "agony columns" of Victorian newspapers. The book was first made available at the APL84 Conference in Helsinki, Finland, June 11-13, 1984 (sponsored by FinnAPL and ACM SIGAPL).

Dr. Franksen’s accounts of Charles Babbage’s propensity towards the solving of encrypted classified ads in the newspapers—manually—in the mid 1800’s—may give you pause for thought about the likelihood of ever being able to provide absolute protection for anything that a very skilled person may want to acquire. APL solutions are given. ...Order it. You’ll learn about “cyphers” and APL and a whole lot more. And you’ll have fun doing it.
Data Encryption

Encryption for Data Transmission

For data transmission of IBM Confidential Restricted data, Corporate Security Instruction #104A requires that encryption be used “except when transmission is entirely via wire (no microwave or satellite) and the wire remains within an IBM location.”

Any data transmission of Confidential Restricted data to and from your terminal will need to be on encrypted lines. Some lines are already encrypted. To find out if the lines you are using are encrypted, or if you need to have some lines encrypted, call the Help Desk at T/L 695-1234... they can get you in touch with the Line Services group.

Realize that the transmission of data over VNET (through the use of workspaces 1 TRANSMIT or 31 VNET) is not always encrypted.

Encryption for Data Storage

Workspace 10 SECURITY offers functions for encryption and decryption of your APL data. Obviously, running all of your data through any conversion function will cause more compute time to be used. If you plan to make heavy use of these functions, you should plan to run sufficient timing tests so that you can determine if this is a reasonable approach for your particular application.

There are two functions provided for converting your data; they are called “ENCODE” and “DECODE”. Let's take a look at them:

```
  v Z+KEY ENCODE IN;ALF;IO;RL ← Localize Origin and Random Link
 [1] ▼ ENCRYPTS CHARACTER RIGHT ARG, BASED UPON 'KEY' IN LEFT ARG
 [2] ▼ IO+0 ← Set the Index Origin
 [3] ▼ ALF+AV ← Atomic Vector is our alphabet
 [4] ▼ RL+((ALF\KEY)+.x\pKEY) ← Calculate RL value from key
 [5] ▼ Z+,((pALF\IN)?1000000 ← Select a repeatable random number
 [6] ▼ Z+,((pALF\(ALF\IN)+Z ← Add that to the text's index values
 [7] ▼ Z+ALF[Z] ← Select characters from our alphabet
 [8] ▼ Z+(pIN)pZ ← Reshape result to match the input

  v Z+KEY DECODE IN;ALF;IO;RL
 [1] ▼ DECIPHERS CHARACTER RIGHT ARG, BASED UPON 'KEY' IN LEFT ARG
 [2] ▼ IO+0
 [3] ▼ ALF+AV ← Atomic Vector is our alphabet
 [4] ▼ RL+((ALF\KEY)+.x\pKEY
 [5] ▼ Z+,((pALF\IN)?1000000 ← Select a repeatable random number
 [6] ▼ Z+,((pALF\(pALF)+(ALF\IN)-Z ← This time we subtract it from the index values of our text
 [7] ▼ Z+ALF[Z]
 [8] ▼ Z+(pIN)pZ
```

Section 7: Stringent Steps for Stringent Needs
Let's try them out with some familiar text:

```
  TEXT←'OPEN THE POD BAY DOORS, PLEASE, HAL.'
  NEW←'SECRET' ENCODE TEXT
  NEW
  _"OєW'y→$}C3g  ___$z^2gφ!*U_?Ve"*9 ←- That's relatively unreadable....
  Do you think you could guess the original text from this?
```

The output may sometimes look a little bit shorter than the original text, just because there may be some undisplayable characters here, but the input and output are the same length:

```
  ρTEXT
  36
  ρNEW
  36
```

This will convert the encrypted data back to plain text:

```
  'SECRET' DECODE NEW
  OPEN THE POD BAY DOORS, PLEASE, HAL.

...just be sure that you don't lose that key!
```

```
  'SECRETS' DECODE NEW
  _Y|φa_|
  Fv~\_,(_3c△k•_->:iω^gt^8,¬YN  ←- Changing the "key" even trivially will
  produce output that bears absolutely no resemblance whatsoever to the original
  encrypted output.

  'SECRETLY' DECODE NEW
  ]_L^8~¬yZy^|
  \θ32_¬'Z=Qss$yVy _K_  

  'SECRETS' ENCODE TEXT
  i$s$|$dUbWls+X d*G ls
  - \$u[*j^R
  `SECRETLY' ENCODE TEXT
  \U:03+[221]a%c"x":"Lp^I/`
```

Compare this with the equivalent example using the "scrambled" approach, back on page 114... which approach do you think looks more secure?

### Additional Reading

Additional reading on the subject of both scrambling and encryption is available in a paper by Bill Hillman, entitled "Information Security Issues in an APL Application", from the Conference Proceedings of the APL84 Conference in Helsinki, Finland (sponsored by FinnAPL and ACM SIGAPL). The proceedings were published as the APL' Quote Quad, Volume 14, Number 4, dated June 1984.
Shared Variables

Two otherwise independent concurrently operating processors can communicate, and thereby be made to cooperate, if they share one or more variables. Such shared variables constitute an interface between the processors, through which information may be passed to be used by each for its own purposes. In particular, variables may be shared between two active APL workspaces, or between an APL workspace and some other processor that is part of the overall APL system, to achieve a variety of effects including the control and utilization of devices such as printers, card readers, magnetic tape units, and magnetic disk storage units.

In use in an APL workspace, a shared variable may be either global or local, and is syntactically indistinguishable from ordinary variables. It may appear to the left of an assignment, in which case its value is said to be set, or elsewhere in a statement, where its value is said to be used. Either form of reference is an access.

At any instant a shared variable has only one value, that last assigned to it by one of its owners. Characteristically, however, a processor using a shared variable will find its value different from what it might have set earlier.

Consider the following simple example of sharing the variable \( V \) between two users 1234 and 5678:

<table>
<thead>
<tr>
<th>User 1234:</th>
<th>User 5678:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5678 ( \text{SVO} \ ' V )</td>
<td>1234 ( \text{SVO} \ ' V )</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( V+5 )</td>
<td>( V+3 \times V \times 2 )</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
</tr>
</tbody>
</table>

The relative sequence of events in the two workspaces, after sharing, is significant; for example, had that last access of \( V \) by 1234 in the foregoing example preceded the setting by 5678, the resulting value would have been 5 rather than 75.

A given processor can simultaneously share variables with any number of other processors. However, each sharing is bi-lateral; that is, each shared variable has only two owners. This restriction does not represent a loss of generality in the systems that can be constructed, and commonly useful arrangements are easily designed. For example, a shared file can be made directly accessible to a single control processor which communicates bi-laterally with (or is integral with) the file processor itself. In turn, the central processor shares variables bi-laterally with each of the using processors, controlling their individual access to the data, as required.

Section 7: Stringent Steps for Stringent Needs
Access sequence disciplines are also imposed on certain of these variables, although one effect of this was noted; namely, variables like the time stamp accept any value specified, but continue to provide the proper information when used. The discipline that accomplishes this effect is an inhibition against two successive accesses to the variable unless the sharing processor (the system) has set it in the interim.

**Distinguished Names for Controlling Shared Variables**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>System Function with Monadic Usage</th>
<th>System Function with Dyadic Usage</th>
<th>Function or Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVC</td>
<td></td>
<td></td>
<td>Shared Variable Control</td>
</tr>
<tr>
<td>SVE</td>
<td>•</td>
<td>•</td>
<td>Shared Variable Event</td>
</tr>
<tr>
<td>SVO</td>
<td>•</td>
<td>•</td>
<td>Shared Variable Offer</td>
</tr>
<tr>
<td>SVQ</td>
<td>•</td>
<td>•</td>
<td>Shared Variable Query</td>
</tr>
<tr>
<td>SVR</td>
<td>•</td>
<td>•</td>
<td>Shared Variable Retraction</td>
</tr>
<tr>
<td>SVS</td>
<td>•</td>
<td>•</td>
<td>Shared Variable State</td>
</tr>
</tbody>
</table>

For formal definitions of all of the system functions and system variables, refer also to *The APL Language Manual* (GC26-3847) and to *An Introduction to APL2* (SH20-9229).
The Ultimate Step: Creating a Server Machine

First, What Is It?

We'll start with an analogy: You have all used TSIO (whether you realize it or not). TSIO is an “auxiliary processor” (a processor that runs independently of APL) which allows APL to communicate with the system that it’s running on, for the purpose of reading and writing files, sending data to a high-speed printer, etc. It happens that TSIO is written in System/370 Assembler code, but it could be written in any language. It could, for instance, have been written in APL code.

In analogous fashion, a “Server Machine” is an APL account which keeps an APL function running all of the time, looking for incoming shared-variable offers which might be made from other users’ accounts.

Let's consider a scenario:

Imagine that you have this special application set up and running under your account number. Phil in Punxsutawney needs to read some data from your database, but you know Phil, and you know that he may try to burrow into more than he is authorized to see if you give him access to it. So instead, you give him a workspace which shares a variable with your account; meanwhile, your account has been signed on and waiting for him to use it since last winter. When Phil finally does share a variable with your account (using user-to-user shared variables), the only thing that he assigns to the shared variable is a short character string which constitutes a command for your application. Your account then looks at that command, looks up his account number in a table to make sure that he is an authorized user, and looks up the information for him. Your functions would have to have established some sort of command language to use for letting Phil order what he needs. Notice that the inquiry is taking place in your workspace, not his. (Notice also that you are paying for it.)

Since the functions that are reading the file aren’t even in Phil’s workspace, there is no chance of him modifying them and thwarting your security checks. When you reduce the data to just the portion that he should see, you simply feed it back to him through the same shared variable.

Why You Might Want to Use This Approach

You may want to consider this sort of approach when you have an application which has various types of data in one file, and where you may want to limit some users to only seeing some of that data. If you do all of the data manipulation within their workspace, consider the limitations: the function that they call must be a single stand-alone function with no sub-routines (or else they could modify some of the sub-functions). Also, the file would have to be opened each time that the user made any inquiry. That’s very time-consuming and wasteful; but the alternative is to have the shared variables be global variables in the workspace, and if that’s the case, there’s nothing to prevent a knowledgeable user from simply assigning his own values to the
control variable and reading the file directly. At that point, all bets are off regarding security.

If you are in this sort of position in the design of an APL application, then maybe you should consider a Server Machine.

We only say "maybe" because the creation and use of a server machine is not without a price (...what is?), and the price just may not be worth it for your data... only you can answer that one. Also realize that this step is a large one, and requires an in-depth knowledge of APL to even consider pursuing it. All of the details will not be spelled out here.

**Why You Might NOT Want to Use This Approach**

You should consider what it implies in the way of resources to run such a machine. In order to offer service to other users through a server machine, you need to have the server machine account signed on and available anytime that any of your users might want to use it. Also, a separate server machine would have to be run on each of the four APL systems here in Kingston if your users are going to be able to access it from whichever machine they happen to sign on to. [Yes, we realize that the lack of user-to-user shared variables across systems is a limitation here, and we are working to remove that restriction. When that's finally corrected, our goal of presenting a "single system image" to you will be resolved.] The responsibility of starting and stopping the server machines each day would be up to you, and moreover, their connect and compute time would be billed to you.

However, since all of the data manipulation is being done in your own workspace (rather than your user's workspace), the security of this approach can't be beat.

...It's just not for everyone.

Since the operation of a server machine requires extra accounts to be signed on all of the time (on each of several systems), you might realize that we here in APL Support are the most likely candidates for using such a facility... and as yet, we haven't taken that step on any of our public offerings. It is a valid approach, and it certainly can be used by any of you. But you do need to consider the costs and implications.

On the next page is a sample server machine. It's a bare-bones model; it does work, but in actual production, you would want to construct a much fancier machine. This one is shown here for instructional purposes only.

"They Also Serve Who Only Stand And Wait."

—John Milton, from *On His Blindness* [1652]
A Sample Server Machine

1. **START**: X, SVLIST, SYNAMES
2. **START UP A SHARED VARIABLE SERVER**
3. **I/O+1**
4. **LOOP**: INITIALIZE ⏤ Open files, set variables, etc.
5. **OFFERS**: RETRACTS ⏤ Look for retracted variables.
6. **OFFERS**: PROCESS SETS ⏤ Look for newly offered variables.
7. **OSV+3600** ⏤ Wait for a shared variable event such as a retract, offer, or set.
8. **X+OSV**
9. → **LOOP**

**INITIALIZATION**

1. **PUT ANY INITIALIZATION CODE HERE: OPEN FILES, SET VARIABLES, ETC.**
2. **SYNAMES+"A",10 p'ABCDEFGHI'** ⏤ Set up list of possible names.
3. **SYLIST+SYNAMES,10 10p'0'** ⏤ List of names and account numbers.
4. **RETRACTS; VARS; B; X** ⏤ Who has retracted?
5. **X+EX B+VARS** ⏤ Retract and erase variable.
6. **SYLIST+(-B)*SYLIST** ⏤ Remove variable name and account number from list.
7. **OFFERS; ACCTS; VARS; X; I; J; NAME** ⏤ Get list of variables offered by one user.
8. **ACCTS+SVQ:0** ⏤ Get list of accounts which made offers
9. **I+0**
10. **L1:I+I+1** ⏤ Loop through list of account numbers...
11. **VARS+SVQ ACCTS[I]** ⏤ Get list of variables offered by one user.
12. **J+0**
13. **L2:J+J+1** ⏤ Loop through list of variables offered by each account.
14. **NAME+SYNAME** ⏤ Get a unique name for shared variable.
15. **NAME+SYLIST** ⏤ Ignore this offer if no name is available.
16. **NAME+SYLIST[I] SVNAME, ' ', VARS[I]** ⏤ Fulfill the offer.
17. **NAME+SYLIST[I] SVNAME, (10p'0')+ACCTS[I]** ⏤ Set shared variable interlock.
18. **L2**

**R+SYNAME; B**

1. **PICK A UNIQUE SHARED VARIABLE NAME**
2. **B+R/SYNAMES'=SVLIST[I];2**
3. **R+B+SYNAMES**

**Z+SETS; S**

1. **RETURNS LIST OF SET VARIABLES**
2. **S+SVS SYLIST[I];2** ⏤ Get status of all shared variables.
3. **S+5.0 1 0 1** ⏤ Which ones have been set?
4. **Z+5+SYLIST** ⏤ Return list of set variables.

**PROCESS VARS; I**

1. **PROCESS SHARED VARIABLES SET BY USERS**
2. **I+0**
3. **L1:I+I+1**
4. **(I+1)+0**
5. **VARS[I];2, ' +SERVE ', VARS[I];2** ⏤ Read, service, and set variable.
6. **L1**

**Z+SERVE R**

1. **PUT THE CODE TO SERVE THE USER HERE**
2. **Z+pR** ⏤ Just as an example, the shape of the variable is returned.
Additional Reading

If you wish to pursue the topic of Server Machines further, you can find additional thoughts in a couple of papers which have been published internally within IBM on this subject. Neither one of these papers directly addresses the subject of server machines under APLSV, but each of the papers may give you ideas on the subject.

The first paper is by Achim Zink (IBM, Mainz, Germany), and is entitled “An APL Secure Machine Facility under VM/370”, published in the Proceedings of the 1978 IBM Symposium on APL (Volume I, page 363); this symposium was held at Foothills College, Los Altos Hills, California, March 28-30, 1978. The proceedings book may be available in many of the IBM site libraries.

Another such discussion was presented at the APL ITL Meeting in San Jose, California, October 16-19, 1984. The paper was entitled “APL2 Server Machine Using GSVP,” by Marty Ziskind (IBM, Endicott, NY). The paper is available from the author.

10 “GSVP” is the “Global Shared Variable Processor,” which allows shared variables to extend between physically separate systems.

11 Notice that people who write about Server Machines commonly have names beginning with “Z”. Someone could probably do their doctoral thesis on this.
Section 8: Protecting Your Data from Batch

RACF and Batch Jobs

By default, all of the datasets that are created through TSIO are protected from batch access. That is, batch jobs cannot access your TSIO data unless you take specific steps to allow such access. This protection is invoked by the “Automatic Data Set Protection” (ADSP) attribute within the system’s “Resource Access Control Facility” (RACF). ADSP causes all newly-created datasets to be protected unless you explicitly specify otherwise.

If you write an application, and you have a need to use a batch job to access some of your TSIO datasets (for example, to load datasets onto the system), we strongly recommend that you contact us while you are still in the planning stages for such an application. We may be able to suggest alternatives which could simplify your operation (such as, in this case, using VNET or wideband for receiving your data).

Assuming, however, that you decide that you do have some special need for batch jobs, the RACF user-IDs that will be accessing the datasets used in the application will have to be given RACF permission to use them. To do this, `LOAD 1 AIDS` and type `RACFPERMIT`. This function will prompt you for the dataset name and the RACF user-ID. You will also need to determine what kind of access they should have. They may have one of four types of access:

<table>
<thead>
<tr>
<th>Access Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = NONE</td>
<td>No access allowed (except by the owner) (i.e. delete RACF authorization)</td>
</tr>
<tr>
<td>1 = READ</td>
<td>Read access only</td>
</tr>
<tr>
<td>2 = UPDATE</td>
<td>Read/Write access</td>
</tr>
<tr>
<td>3 = ALTER</td>
<td>Read/Write access plus Rename/Delete authorization</td>
</tr>
</tbody>
</table>

Controlling the Security of Applications on Kingston APLSV — The APL Jot*Dot Times — 125
To see who has been previously permitted, use the "RACFLIST" function. Enter the name of the dataset(s) you wish to see. The function will return the RACF user-IDs which can access that dataset and what type of access they have.

```
)LOAD 1 AIDS
SAVED 10.22.11 08/20/84

RACFLIST
>>>DATASET NAMES? (ONE PER LINE)
? ^123456 SHARED.PROJECT
?

RACF AUTHORIZATIONS FOR DSN=123456 SHARED.PROJECT

<table>
<thead>
<tr>
<th>RACF USER ID</th>
<th>ACCESS LEVEL</th>
<th>ACCESS COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A123456</td>
<td>READ</td>
<td>9</td>
</tr>
</tbody>
</table>
```

To change the authorization:

```
)LOAD 1 AIDS
SAVED 10.22.11 08/20/84

RACFPERMIT
>>>DATASET NAME? ^123456 SHARED.PROJECT
>>>RACF USER ID (OF BATCH JOB)? A123456

RACF ACCESS LEVELS:
  0 = NONE (NO ACCESS ALLOWED)
  1 = READ READ ACCESS ONLY
  2 = UPDATE READ/WRITE ACCESS
  3 = ALTER READ/WRITE ACCESS PLUS RENAME/DELETE

>>>ACCESS AUTHORITY (0, 1, 2, OR 3)? 3

*** RACF USER [A123456] AUTHORIZED FOR [ALTER] ACCESS TO DSN=123456 SHARED.PROJECT
```
In case you feel rather like an outsider because of some of the terminology used in this manual, let's try to get together on it. Here are a few of the terms that we often consider to be common-knowledge (but may not be).

[Items marked with “•” also appear elsewhere in this glossary.]

**alpha-numeric:** A sequence of characters consisting of any mixture of letters and numbers (such as “ABC123”).

**APL:** The name of the programming language that is available on Kingston system and discussed in this manual; named for the book entitled “A Programming Language” (K.E. Iverson, published by Wiley and Sons, 1962).

**APLSV:** The particular “brand-name” of APL that’s available on the Kingston system and discussed in this manual; “APLSV” stands for “APL with Shared Variables” (a means of managing data).

**access code:** This is the first letter of the name that displays when you sign on to APL (that’s the “K” in the “KJDOE” name in the sign-on message on page 27.) Contrary to public opinion, it is not there simply for the purpose of misspelling your name; it’s a code that determines which of the telephone lines you are authorized to use when you dial into the system. For more information, “\LOAD 1 PHONES”.

**account number:** This is the numeric portion of your APL sign-on number. By local convention on the Kingston system, it normally (but not necessarily) matches your IBM employee serial number.

**batch processing:** When you sign on to APL, you type in a statement and you get a result printed back at your terminal. This result will help you to decide what you may want to type in next. This is interactive processing. The opposite approach to this is to gather up all of your inputs in one “batch”, and send the entire list of requests to a system for processing. You would then get back your results at some later time. An example of this type of processing is the APL facility called “DEFER” (in workspace 30 DEFER).

**command dataset:** This is a dataset containing only commands and list of the account numbers which are authorized to execute the various commands. This mechanism provides a secure means of allowing scattered users to access your data in a controlled fashion. The concepts behind this scheme is discussed in detail on pages 85-91, and the functions which allow you to maintain command datasets are described in detail on pages 92-94.

Controlling the Security of Applications on Kingston APLSV —The APL Jet Dot Times— 183
**compute time:** This is the amount of time that APL has actually spent performing work, or doing calculations, for you as opposed to working for the scores of other users who may also currently be signed-on to the same computer. This time is normally a small fraction of the time that you were connected to APL. Printing output at the terminal, for example, takes very small amounts of compute time. [Also called CPU time]

**connect time:** This is the amount of time that your APL account was signed-on to (or connected to) APL. If you sign on at 3 pm and off at 4 pm, you've used an hour of connect time.

**CPU:** CPU stands for Central Processing Unit... it is the part of the computing system that actually does the computing, as opposed to “peripheral” equipment, such as printers and storage devices.

**dataset:** A dataset is an orderly collection of data stored on disk. In APL, it is not the same as a workspace. Refer to page 73 for a complete discussion of this topic.

**DEFER:** See “batch”.

**expunge:** This refers to the dynamic erasure of functions or variables. The “)ERASE” command also provides an erasure facility, but it may not be issued from within a function. “)EX” is a very similar facility, and may be issued from within your code. Refer to page 51.

**file:** See “dataset”.

**function:** The term that APL uses for “program”. Often abbreviated as “fn”.

**hacker:** “The hacker stereotype is a pudgy male with a fish-belly-white complexion who swills soft drinks, lives on candy bars and spends most of his waking hours in front of a terminal, playing games or trying to penetrate Defense Department Networks.” —Time Magazine, May 1983

**Hotline:** Kingston APL offers the convenience of having a single phone number for getting in touch with anyone in APL Administration, programming, and management. If you know precisely who to call, you can save a step by calling them directly. But if you're ever not sure who can answer your questions, just call the APL Hotline, T/L 695-1234 (or outside, on 914-385-1234). [Also known as the Help Desk]

**IC dataset (Indirect Command dataset):** See “command dataset”.

**interactive:** The opposite of batch... see “batch”.

**latent expression:** [DLX] Causes a workspace to “come up running” when it’s LOADED. Refer to The APL Language Manual (GC26-3847) for details of usage, and see the discussion in this manual on page 48.

**library:** The set of workspaces that you have saved is called your library. Each workspace is identified by your account number and the name that you assign to it. In referring to workspaces in your own library, the account identification may be omitted; your own identification is then supplied automatically.

**problem number:** This is your APL billing number. Each group, typically a department, has a number assigned, which groups the charges together under one bill. The problem number is six characters long, mixed alphanumeric. It is in the form of “123XAK”. The first three characters are usually (but not necessarily) your department number.

Your problem number manager is the manager who receives the billing for your APL usage. Most often (but not necessarily), he is the department manager. Both your problem number and your problem number manager are listed in workspace 1 ACCOUNT.
**RACF:** “Resource Access Control Facility”; this is a security package offered with the host system (MVS), which controls who can get to each of the datasets on the system.

**reserved dataset:** This is a dataset which may only be accessed by its owner or by users specifically authorized by the owner. A reserved dataset is a secure means of storing sensitive data. Refer to pages 85-91.

**right-paren:** The right parenthesis—or closing parenthesis—“)” is always used to preface a system command; used when signing on [as in “)123456:ABCDEF”] or “)OFF”, or to “)LOAD” or “)SAVE” a workspace.

**system command:** An APL system recognizes two broad classes of instructions, “expressions” and “system commands”. System commands control the initiation and termination of a work session, saving and reactivating copies of a workspace, transferring data from one workspace to another, and a variety of tasks within the workspace.

System commands can be thought of as being a sort of link to the outside world, allowing the users of APL functions and variables access to the environment which may not be defined as properly being a part of the APL Language.

System commands can be invoked only by individual manual entries from the keyboard and cannot be executed dynamically as part of a defined function. They are distinguished from APL statements in that they are always prefixed by a right (closing) parenthesis.

**TSIO:** TSIO stands for “Time-Shared Input/Output”; it’s an “auxiliary processor” (a processor that runs independently of APL) which allows APL to communicate with the system that it’s running on, for the purpose of reading and writing files, sending data to a high-speed printer, etc.

**variable:** In APL, an assignment of data to a name creates a variable, so named because its value may be varied at any time by simply reassigning it. For example, “T+‘SOME TEXT’” stores nine characters under the name “T”, and “E+2+2” stores the result of a calculation under the name “E”. These values are then stored in the APL workspace.

**workspace:** The common organizational unit in an APL system is the workspace. When in use, a workspace is said to be “active”, and it occupies a block of working storage in the central computer. Part of each workspace is set aside to serve the internal workings of the system, and the remainder is used, as required, for storing items of information and for containing transient information generated in the course of a computation.

A terminal always has an “active workspace” associated with it during a work session, and all transactions with the system are mediated by it. In particular, the names of “variables” (data items) and “defined functions” (programs) used in calculations always refer to objects known by those names in the active workspace; information on the progress of program execution is maintained in the “state indicator” of the active workspace; and control information affecting the form of output is held within the active workspace.

Inactive workspaces are stored in “libraries”, where they are identified by arbitrary names. They occupy space in secondary storage facilities of the central computer and cannot be worked with directly. When required, copies of stored workspaces can be made active, or selected information may be copied from them into an active workspace.

**ws:** Abbreviation for “workspace”, or shown as “wss” for “workspaces”.

---

Controlling the Security of Applications on Kingston APLSV —The APL Job Data Times—
**WS FULL:** "Workspace full"—Error message indicating that the workspace is filled, perhaps by temporary values produced in evaluating a multiple-step expression, or by values of shared variables.

1. Clear the state indicator
2. Erase unneeded objects
3. Revise calculations to use less space
4. Rewrite application to use external files (TSIO datasets) for data storage

For further information on APL error reports, refer to The APL Language Manual (GC26-3847).
Acknowledgements

This manual was written by Jon McGrew, with material gathered in part from previous versions of The APL User's Guide and The APL Language Manual, by Adin Falkoff and Ken Iverson. Additional material for this manual was contributed by Howard J. Smith, Jr. and Barbara Herrick. Some of the descriptions of security features were previously published by Jon McGrew in "The APL Jot Dot Times." Descriptions of facilities which are offered in Public Library workspaces on the Kingston system were written by Jon McGrew, Evan Jennings, Mike Harellick, Blair Martin, and Joey Tuttle. Some of the APL system security measures described here were designed and implemented by Richard H. Lathwell, Joey Tuttle, and Bruce Hartigan. Thanks also to Evan Jennings and Mike Van Der Meulen for repeated proof-reading of this newsletter.

References


[8] Unnamed, "IBM Information Classification and Control," Form No. ZZ00-0001, IBM Corporation, December 1979
Index

access code
  definition 183
access controls 85, 137, 138
  classified information 85, 136, 141
  computing installations 157
  supplier of services 154
command datasets 85
“confined” accounts on APL 43
field level 149
histories of access 101, 146, 175, 176
  Registered IBM Confidential (RIC) 178, 179
physical access 156
  computing installations 155
  personal computers 147
  small systems and input/output devices 158
  supplier of services 154
preventing repetitive attempts 27, 149
record level 149
reserved datasets 85
supplier of services 154
user of services 85, 152
access reports
  Registered IBM Confidential (RIC) 146, 157
access to APL
  for dial-up terminals 19
  See also information inside front cover
  bad connection 23
access to data
  repetitive attempts 149
  shared 85, 97
accountability
  owner not accountable for user 147
  portable storage media inventory 158, 159
  shared sign-on passwords 25, 32, 139
  user requirements 152
  use of your account 32
accounting information (DAI) 17
example of use  6
account number
   definition  183
account number (APL sign-on number)  24, 47
Accounts Payable  137
Accounts Receivable  137
acknowledgements  187
ACM (The Association for Computing Machinery)  70, 115, 117
active workspace  37, 185
Adams, Douglas  112
Administration
   password change requirements  140
ADSP  125
after-the-fact investigations  101, 146
1 AIDS (workspace)
   ALLOCATE function  79
   for classifying files  79
   IC function  85, 92
   inner workings (for programmers)  96
   INITIALIZE function  80
   RACF functions  125
ALLOCATE function  79
alpha-numeric
   definition  183
alter access
   controlling  136
ambivalence  32
ambi-valence  71
ampersand “&”  86
“An Introduction to APL2” (manual)  ii, 187
anagrams  113
announceements
   of changes  5
annual reviews
   See periodic reviews
aperture cards
   See micrographics
APL
   definition  183
“The APL Language Manual”  2
APL functions
   local  58, 59
   locked  52, 57
APL Hotline  3
APLSV
   definition  183
“The APLSV Version 3 User’s Guide”  2
application
   definition  148
   for a new account  30, 151
application design
   owner of data  148-149
   required functions  149-150
   security requirements  148, 149
application programming  128, 137
application programs
  classifying 136
applications
  common
    owner's responsibilities 148
internal
  compliance 148
  exemptions 148
new
  final testing 149
stable
  exemptions 148
testing
  Confidential data 149
  owner's requirements 148
appraisal information 172, 173
AP124
  full-screen auxiliary processor
    display inhibit 35-36
archived datasets 77, 83
Association for Computing Machinery 70, 115, 117
attention 53, 62
Auberson, David 29
auditability requirements 146
definition 146
  annual requirement 148
  application systems 161
  non-IBM terminals 143
  owner's responsibilities 133, 147
  risk assessment and risk acceptance 144
  satisfactory compliance 146
  supplier of services 154
audit functions 137
audits 16, 128
  Registered IBM Confidential (RIC) 178, 180
audit trails
  control elements 146
  restricted utilities 146
  TSIO dataset usage 101
authorization records
  for access to a computing installation 155
Automatic Data Set Protection
  See ADSP
auxiliary processor
  server machine 120
auxiliary processor
definition 185
  full-screen processor (AP124)
    display inhibit 35-36
avoidance
  of software controls 138, 146
    through repetitive attempts 149
awareness
  responsibility 132, 163, 165
Babbage, Charles 115
backup procedures 138
backup procedures for data 10
badges 156
basic responsibilities 132-134
management 132
owner of data 133
supplier of services 134
user of services 134
batch
job submission 145
batch jobs 125
batch processing
definition 183
bi-annual audits 16
billing 14, 137
bindings
obscuring classification 170
bistromathics 112
blotting-out passwords 21, 34, 139, 149, 157
blueprints 171
bombs
See workspace 10 SETBOMB
books 169
brackets
round
See right-paren
bucks
elephant 69
bursting/collating rooms
near computing installations 155
business controls 138
business impact
unacceptable
risk assessment and risk acceptance 154
cancellation of service confirmation of 154
canonical representation (CR) 53
carbon paper 158
carbon ribbon
  precautions in using 34
cards
  punched 169, 171
career path information 172
cartridge
  mass-storage system
    control of 158
cassette tapes
  control of 158
  1 CATALOG (workspace)
    for locating workspaces 39
central processing 128
change histories 146
changes
  keeping current 5
classification
  access controls 136
  application programs 136
  by the owner 157
  default 81
documentation 136
  IBM Confidential (IC) 172, 174
    definition 172
  IBM Confidential-Restricted (ICR) 175, 176
    definition 175
  identification 170
    on input/output 135
  Internal Use Only (IUS) 171
    definition 171
  levels 135, 169
    sign-on passwords 33
    supported on Kingston APLSV 4, 81
micrographics 136
  of assets 145
  of information assets 133
  of output 135, 170, 171
    for control elements 139
    for restricted utilities 139
    required for new applications 150
  of TSIO datasets
    how to do it 79
preprinted 170
Registered IBM Confidential (RCL) 178, 180
  definition 178
  responsibility 169
software 136
classification and control of information by the owner 157
changing supplier of services 154
identification 170
responsibilities 135, 136, 169
assigning 132
classified information access controls 136
supplier of services 154
“classified information” labels 159
CLASSIFY function 81
clear text
See cryptography
See scrambling of data
code names
unannounced products 171
collusion 146
colon “:” 39
COM installations near computing installations 155
command system
See system command
COMMAND DISALLOWED message
restricted library 42
command datasets 92
definition 85, 183
inner workings (for programmers) 95
overview 85
symbolic parameters 86
common applications
owner’s responsibilities 148
communication of controls to affected parties 148
of facilities and practices
owner’s responsibilities 148
supplier of services iii, 155
workspace 1 NEWS 5
comparison tolerance (OCT) 45
localizing 45
compatibility with IBM products 141
compliance
causing unacceptable business impact 154
demonstrating 146
internal applications 148
monitoring
Corporate IS&A Staff responsibility 164
Corporate Security Staff responsibility 162
delegation of authority 147
non-IBM terminals 153
organizational unit responsibility 165
owner’s requirements 148
supplier of services 154
user's requirements 151
non-IBM terminals
annual reviews 142
technical guidance 160
satisfactory
definition 145
auditability requirements 146
demonstrating 146
self assessment and planning
annual reviews 144
stable applications
review 148
user must certify 151
user's requirements 134
using IBM products 137
compliance monitoring 133
Corporate IS&A Staff responsibility 164
Corporate Security Staff responsibility 162
management's requirements 144
non-IBM terminals 153
annual reviews 142
technical guidance 160
organizational unit responsibility 165
owner of data
delegation of authority 147
owner's requirements 148
records for 146
risk assessment and risk acceptance 144
self assessment and planning
annual reviews 144
supplier of services 154
user's requirements 151
compromise
of passwords 32, 140
computer access telephone numbers
classifying 158
don't post 153
protecting 142
computer, personal
See personal computer
compute time 47
definition 184
computing installation
definition 155
computing installations
physical access controls 155
user of services 155
concentrations of data 170, 175, 178
"condition of continued employment"
Corporate Security 128
confidential disclosure agreement 136, 142, 161
Confidential (IC)
definition 172
disclosure outside of IBM 174
requirements for 172
risk assessment and risk acceptance 145
Confidential-Restricted (ICR)
definition 175
deletion of data 83
disclosure outside of IBM 177
overwriting prior to deletion 83
requirements for 175
risk assessment and risk acceptance 145
terminal encryption 116
confined accounts 43
connect time 47
definition 184
consistency of control 145
contents 130, 168
CONTINUE workspace 41
created automatically by the system 40
restricted library 42
continued operation 155
continuity of control 145
contractors
within a computing installation 156
control centers
near computing installations 155
control elements
definition 137
access controls 157
auditability 146
controls 138
on portable storage media 159
owner's responsibilities 148
requirements 137-139
risk assessment and risk acceptance 145
control facilities 132, 134
documentation of
owner's responsibilities 148
supplier of services iii, 155
control functions 137
controlling events 62-71
copy access
controlling 136
copying
IBM Confidential-Restricted (ICR) 176
Internal Use Only (IUO) 171
Registered IBM Confidential (RIC) 179, 180
"Corporate Security Instruction #104A" (document) 128-166
See also specific topics
format 128
ordering 2, 127
purposes 128
scope 128
table of contents 130
Corporate IS&A Staff responsibilities 163
Corporate Security Staff responsibilities 162
costs
statements of 172, 175
CPU
definition 184
   time
      definition 184
creating IC datasets 92
credit cards 32
cryptography 141
definition 141
custodial responsibilities
data 128, 132
customer engineers
   within a computing installation 156

data
   See datasets
   See variables
data access
   histories of access 146, 175, 176
      Registered IBM Confidential (RIC) 178, 179
   repetitive attempts 149
   supplier of services
      delegation of authority 154
data encryption 116
data entry 128
data entry areas
   near computing installations 155
data files
      See written procedures
data integrity 97
data protection 128
data reduction programs 138
data scrambling 113
DATASET NOT FOUND message 104
dataset privacy
   inner workings (for programmers) 95
datasets
      definition 184
      command 85
         definition 85
         inner workings (for programmers) 95
         symbolic parameters 86
      creating 79
      how to classify them 79
   IC
      creating and maintaining them 92
keeping track if them 73
reserved 85
definition 85, 185
   inner workings (for programmers) 95
temporary versus permanent 77
dates
current 47
protecting 175
Davenport, T. F., Jr. 129, 187
day of the month (current date) 47
deactivation
See termination of business needs
deactivation of passwords
   by supplier of services 157
   by user 157
   supplier of services 16, 157
   user requirements 152
deal function (n?n) 111
DECODE function
   for data decryption 116
decryption
   See cryptography
   See scrambling of data
DEFER
definition 184
del “v”
   for defining functions 52
delegation of authority
data access
   supplier of services 154
   owner’s responsibilities 147
DELETE function 83
deletion authority
   Confidential data 149
   supplier of services responsibilities 157
deletion of account 154
 (chart) 30
del-tilde “~”
   for locking functions 52, 53
demounting
   portable storage media 159
denial of service 26
 (chart) 30
   supplier of services 154
DESCRIBE 39
design of applications
   owner of data 148-149
   required functions 149-150
   security requirements 148, 149
detection
   of violations 137, 138
   supplier of services 154
development of applications
   security requirements 148, 149
dial-up terminals
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-minute time-out at sign-on</td>
<td>27</td>
</tr>
<tr>
<td>Telephone access</td>
<td>19</td>
</tr>
<tr>
<td>Lesbian connection</td>
<td>23</td>
</tr>
<tr>
<td>Dial-8 network</td>
<td>19</td>
</tr>
<tr>
<td>See also information inside front cover</td>
<td></td>
</tr>
<tr>
<td>Digital document transmission</td>
<td>128, 143</td>
</tr>
<tr>
<td>Cryptography</td>
<td>141</td>
</tr>
<tr>
<td>IBM Confidential (IC)</td>
<td>173</td>
</tr>
<tr>
<td>IBM Confidential-Restricted (ICR)</td>
<td>176</td>
</tr>
<tr>
<td>Registered IBM Confidential (RIC)</td>
<td>179</td>
</tr>
<tr>
<td>Direction</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td></td>
</tr>
<tr>
<td>Extended term</td>
<td>178</td>
</tr>
<tr>
<td>Operational</td>
<td></td>
</tr>
<tr>
<td>Extended term</td>
<td>175</td>
</tr>
<tr>
<td>Short term</td>
<td>172</td>
</tr>
<tr>
<td>Director of IS&amp;A responsibilities</td>
<td>162</td>
</tr>
<tr>
<td>Director of IS&amp;A Staff responsibilities</td>
<td>163</td>
</tr>
<tr>
<td>Director of Security responsibilities</td>
<td>162</td>
</tr>
<tr>
<td>Disaster recovery planning</td>
<td>12, 138</td>
</tr>
<tr>
<td>Disclosure outside of IBM</td>
<td></td>
</tr>
<tr>
<td>IBM Confidential (IC)</td>
<td>174</td>
</tr>
<tr>
<td>IBM Confidential-Restricted (ICR)</td>
<td>177</td>
</tr>
<tr>
<td>Internal Use Only (IUC)</td>
<td>171</td>
</tr>
<tr>
<td>Registered IBM Confidential (RIC)</td>
<td>181</td>
</tr>
<tr>
<td>Disconnected service machine</td>
<td>120-123</td>
</tr>
<tr>
<td>Diskettes</td>
<td>171</td>
</tr>
<tr>
<td>Control of</td>
<td>158</td>
</tr>
<tr>
<td>Security</td>
<td>135, 170</td>
</tr>
<tr>
<td>Disk pack</td>
<td></td>
</tr>
<tr>
<td>Control of</td>
<td>158</td>
</tr>
<tr>
<td>Display inhibit</td>
<td>35</td>
</tr>
<tr>
<td>Disposal of residual information</td>
<td>83, 143, 150</td>
</tr>
<tr>
<td>Distinguished names</td>
<td></td>
</tr>
<tr>
<td>For controlling shared variables</td>
<td>119</td>
</tr>
<tr>
<td>Distributed processing</td>
<td>128</td>
</tr>
<tr>
<td>Distribution of documents</td>
<td></td>
</tr>
<tr>
<td>Electronic</td>
<td>128, 143</td>
</tr>
<tr>
<td>IBM Confidential (IC)</td>
<td>173</td>
</tr>
<tr>
<td>IBM Confidential-Restricted (ICR)</td>
<td>176</td>
</tr>
<tr>
<td>Registered IBM Confidential (RIC)</td>
<td>179</td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Classifying</td>
<td>136</td>
</tr>
<tr>
<td>Of facilities and practices</td>
<td></td>
</tr>
<tr>
<td>Owner's responsibilities</td>
<td>148</td>
</tr>
<tr>
<td>Supplier of services</td>
<td>iii, 155</td>
</tr>
<tr>
<td>Document distribution</td>
<td></td>
</tr>
<tr>
<td>Electronic</td>
<td>128, 143</td>
</tr>
<tr>
<td>IBM Confidential (IC)</td>
<td>173</td>
</tr>
<tr>
<td>IBM Confidential-Restricted (ICR)</td>
<td>176</td>
</tr>
<tr>
<td>Registered IBM Confidential (RIC)</td>
<td>179</td>
</tr>
<tr>
<td>Documents</td>
<td>169</td>
</tr>
<tr>
<td>Document #104A</td>
<td></td>
</tr>
<tr>
<td>See “Corporate Security Instruction #104A”</td>
<td></td>
</tr>
</tbody>
</table>
edit functions 137
education
  employees 132, 163, 165
electronic document distribution 128, 143
IBM Confidential (IC) 173
IBM Confidential-Restricted (ICR) 176
Registered IBM Confidential (RIC) 179
elephant bucks 69
employees
distribution limited to 172, 174, 175, 176, 178, 180
education 132, 163, 165
non-regular or non-IBM 32, 137
  "confined" accounts on APL 43
  Open House or Family Day 43
  within a computing installation 156
employment
  condition of continuing
    Corporate Security 128
empty vector
  response to prompts 68
ENCODE function
  for data encryption 116
encrypted data
treated as unclassified 159
encryption
  See also cryptography
  See also scrambling of data
  for IBM Confidential (IC) 173
  for IBM Confidential-Restricted (ICR) 116, 176
  for Registered IBM Confidential (RIC) 179
keys 138, 141, 143, 159
  classification 152
  classifying 141
  compromise 152
disclosure 152
generating 141
length 141
  recording 152
terminal display 152
terminal entry 152
user requirements 152
encryption keys 138, 141, 143, 159
changes/deletion/disposal 146
classification 152
classifying 141
compromise 152
disclosure 152
generating 141
length 141
recording 152
terminal display 152
terminal entry 152
user requirements 152
engineering services 171
enqueue/dequeue facility 97-100
entry
data 128
envelopes
for IBM Confidential (IC) 173
for IBM Confidential-Restricted (ICR) 175
for Internal Use Only (IUO) 177
for Registered IBM Confidential (RIC) 179
equipment controls
supplier of services 154
erase
dynamic 51
erasing files 143
See also DELETE function
error
examples of dealing with 62-71
handling 62-71
report 186
side-tracking 66-71
trapping 62-71
escorting of visitors
within a computing installation 156
essential operation 155
event handling 62
excuses 2
execute "x"
dyadic 66-71
monadic 64-65
execute alternate (EA) 66, 67
execution authority
Confidential data 149, 150
restricting 138, 147
supplier of services responsibilities 157
exemptions
internal applications 148
exposure
of encryption keys 142
exposures
identifying 144
risk assessment and risk acceptance 144
in program products 145
reporting 132, 134
owner's responsibilities 147
unauthorized personnel 155
expunge (EX) 49, 51
definition 184

facilities
documentation of
owner's responsibilities 148
supplier of services iii, 155
facsimile transmissions
of IBM Confidential (IC) 173
of IBM Confidential-Restricted (ICR) 176
of Registered IBM Confidential (RIC) 179

FAILURES function 104
Falkoff, A. D. 187
Family Day 43
field-level access control 149
FILELIB (workspace) 73
CLASSIFY function 81
DELETE function 83
for classifying files 79
for identifying files 73
files
See datasets
See written procedures
films 169
final testing
new applications 149
FinnAPL 115
Fix function (EX) 56
dyadic form 57
floppy disks 171
control of 158
security 135, 170
FOC function 60, 67
forwarding stations
messages 143
frame rooms
restricted access 156
Franksen, Ole 115
full-screen processor (AP124)
display inhibit 35-36
function establishment (EX) 56
dyadic form 57
functions
definition 184
ambi-valent
definition 71
local 58, 59
locked 53
sub-functions 54
uninterruptible 69

general requirements 132
owner of data 147
supplier of services 154
user of services 151
generating data
IBM Confidential (IC) 173
IBM Confidential-Restricted (ICR) 175
Internal Use Only (IUO) 177
Registered IBM Confidential (RIC) 178
Gerroll, David 29
global variables 54
grace period
for changing passwords 28
Graham, Alan 70
guidance
technical
supplier of services 160
hackers 19
definition 184
Haldeman, H. R. 62
hard-copy output
classifying 135, 170, 171
control elements 139
required for new applications 150
restricted utilities 139
Harellick, Mike 187
Harie 29
Hartigan, Bruce 187
hash total programs 138
Herrick, Barbara 187
hiding passwords 21, 34, 139, 149, 157
Hillman, Bill 117
histories
of access 146
access controls for ICR data 175, 176
access controls for RIC data 178, 179
workspace 1 SMF 101
of changes 146
Hodges, A. G. 62
home terminals 142
supplier of services responsibilities 160
user requirements 153
Hotline 3
definition 184
hour (current time) 47
HYZAP 138

IAS executive/manager responsibilities 164
"IBM Information Classification and Control"
document 168-181
See also specific topics
ordering 167
IBM Confidential (IC)
definition 172
disclosure outside of IBM 174
generating 173
mailing 173
requirements for 172, 174
risk assessment and risk acceptance 145
telephone calls 174
transmitting 173
tavel with 174

IBM Confidential-Restricted (ICR)
definition 175
copying 176
deletion of data 83
disclosure outside of IBM 177
generating 175
mailing 175
overwriting prior to deletion 83
requirements for 175, 176
risk assessment and risk acceptance 145
telephone calls 176
terminal encryption 116
transmitting 176
tavel with 177

IBM Family Day 43
IBM PC
See personal computer
IC function 85, 92
inner workings (for programmers) 96
IC (classification)
See IBM Confidential
IC datasets
definition 85
creating and maintaining them 92
overview 85
symbolic parameters 86
ICR (classification)
See IBM Confidential-Restricted
"identifiable to an owner" 157
identification
badges 156
of assets 145
of classification 170
of information assets 133

IEHDASDR 138
image protection 128
impact
unacceptable
risk assessment and risk acceptance 154
implementation of applications
owner of data 148-149
required functions 149-150
security requirements 148, 149
importance of assets
judging 133
improbable events 63
IMPROPER LIBRARY REFERENCE
attnpting to LOAD a workspace from a locked account 26
attnpting to LOAD CONTINUE 41
confined (non-IBM) users  43
restricted library  42
improprieties
  See misuse
INCOMPLETE SIGN-ON message  24
independent reviews
  by management  139
index origin (IO)  110
  localizing  45
indirect command datasets
  definition  85
  creating and maintaining them  92
  overview  85
  symbolic parameters  86
indirect commands
  inner workings (for programmers)  95
information assets
  supplier of services  154
Information Asset Security responsibilities  164
information classification and control
  reclassifying
    supplier of services  154
  responsibilities  135, 136
    assigning  132
Information Systems (I/S)
  application design  148
INITIALIZE function  80, 83
input
  keyboard  55, 66
input/output
  classifying  135
  physical access controls  158
input/output areas
  near computing installations  155
inquiry systems  148
installation
  computing
    definition  155
installation rules
  documentation of
    owner's responsibilities  148
    supplier of services iii, 155
Instruction #104A
  See "Corporate Security Instruction #104A"
intangible forms  171
integrity
  data  97
interception of transmissions  141
internal applications
  compliance  148
  exemptions  148
Internal Use Only (I/O)
  definition  171
    copying  171
    disclosure outside of IBM  171
generating 171
mailing 171
requirements for 171
risk assessment and risk acceptance 145
transmitting 171
travel with 171
interrupts 53, 62, 69
See also error trapping
“An Introduction to APL2” (manual) ii, 187
inventories
portable storage media 159
auditability requirements 147
Inventory Control 137
ITPS transmissions
of IBM Confidential (IC) 173
of IBM Confidential-Restricted (ICR) 176
of Registered IBM Confidential (RIC) 179
IWO (classification)
See Internal Use Only
Iverson, K. E. 183, 187

JCL 86
Jennings, Evan 187
job submission 145
“Jot® Dot Times”
engravings i (title page)
history ii
production notes i (title page)
Katzenbach, N. deB. 129, 187
dey 39
deyboard input
   prompting 55, 66
deyboard-unlock time 47
deying time 47
deys
   See also passwords
decryption 138, 141, 143, 159
dehanges/deletion/disposal 146
classification 152
classifying 141
decompromise 152
deisclosure 152
generating 141
dehgth 141
derecording 152
terminaldisplay 152
terminalentry 152
user requirements 152

Kindler, H. S. 63

labels
   “Property of IBM” 159
dast-used date
   at sign-on time 27
detent expression ([OX]) 47, 48
definition 184
dexample of use 7

Lathwell, Richard H. 187
degal remedies 128, 146
delength
   of encryption keys 141
   of passwords
      sign-on 28, 140

delevels
   of classification 135, 169
   sign-on passwords 33
   supported on Kingston APLSV 4, 81
libraries 37, 39, 184, 185
  public 39
  restricted 42
tape/disk
  near computing installations 155
library
  definition 184
library functions 128
life
  meaning of 52
"Life, the Universe and Everything" (novel) 112
Lincoln, A. 62
line counter (LC) 47
list programs 148
)LOAD command
  concealing 34
  with a password 39
load-balancing 22
load-only workspaces 42
local functions 58, 59
localization
  of functions 58, 113
  of system variables 45-46, 113
  LOCALIZE (workspace)
    for localizing functions 59
lock 39
  physical 171, 173, 176, 180
locked functions 52, 53, 57
locking accounts 26
  (chart) 30
  supplier of services 154
locking cabinets 171, 173, 176, 180
locking workspaces 39
lock words
  See passwords
log files 138
  for sensitive programs 138, 146
logging
  of sign-on attempts 27
logging functions 137, 138
  supplier of services 154
logging of data access 102
logging off
  terminals
    user responsibilities 153
loss of information assets 128, 132, 137
  Registered IBM Confidential (RIC) 180
  sensitive programs 138
  through collusion 146
mag cards 169, 171
  control of 158
  security 135, 170
magnetic tapes 169, 171
Mailbox facility (on-line) 6
  See also workspace 1 NEWS
mailing
  IBM Confidential (IC) 173
  IBM Confidential-Restricted (ICR) 175
  Internal Use Only (IUO) 171
  Registered IBM Confidential (RIC) 179
mail to users and managers 13
main storage
  clearing 143, 150
management
  options 128, 171
  “management-approved purposes” 4, 132, 134, 142, 153, 155
management responsibilities
  basic 132
management supervision 137
  during testing 149
Manager’s Manual 171
Martin, Blair 187
mass-storage system cartridge
  control of 158
  “may contain classified information” labels 159
McGee, F. 67
McGrew, Jon 187
McGurk’s Law 63
Meaning of Life 52
messages and text 143
messenger
  shooting 2
micrographics
  classifying 136, 170, 171
microwave transmission 116, 141, 176, 179
millisecond (current time) 47
Milton, John 121
minimum security requirements 128
minute (current time) 47
misuse of information assets 128, 132, 137
  sensitive programs 138
  through collusion 146
MIX function
  for data scrambling 113
models 171
modification
  Confidential data 149
  of software controls 138
modification authority
for production code
approval for 138
restricting 138, 147
supplier of services responsibilities 157
modular code
warning against 54
modules
object code 149
monitoring compliance 133
Corporate IS&A Staff responsibility 164
Corporate Security Staff responsibility 162
management's requirements 144
non-IBM terminals 153
annual reviews 142
technical guidance 160
organizational unit responsibility 165
owner of data
delegation of authority 147
owner's requirements 148
records for 146
risk assessment and risk acceptance 144
self assessment and planning
annual reviews 144
supplier of services 154
user's requirements 151
month (current date) 47
mounting
portable storage media 159
"Mr. Babbage's Secret: The Tale of a Cypher—and APL" 115
multiple-user systems
controlling access to data 147, 157
alternatives 147

name classification (\textit{NC}) 71
names
conflicts 69
distinguished
for controlling shared variables 119
Nanette 71
need to know
auditability requirements 146
data access passwords 140
encryption keys 142
sign-on passwords 140
use of IBM Confidential (IC) 172, 173, 174
| use of IBM Confidential-Restricted (ICR) | 175, 176 |
| use of Registered IBM Confidential (RIC) | 178, 180 |
| network/system control centers | |
| near computing installations | 155 |
| NEW PASSWORD UNACCEPTABLE message | 28 |
| new applications | |
| final testing | 149 |
| required functions | 149-150 |
| security requirements | 148, 149 |
| NEWS (workspace) | 5 |
| sending your own messages | 6 |
| NO APL PORTS AVAILABLE message | 24 |
| no message at sign-on time | 23 |
| non-compliance | |
| by a user | |
| owner's accountability | 147 |
| documenting | 144, 146 |
| evaluating | 144 |
| identifying instances | 144 |
| monitoring | 133 |
| Corporate IS&A Staff responsibility | 164 |
| Corporate Security Staff responsibility | 162 |
| organizational unit responsibility | 165 |
| non-IBM terminals | |
| annual reviews | 142 |
| technical guidance | 160 |
| self assessment and planning | |
| annual reviews | 144 |
| non-disclosure agreement | 136, 142, 161 |
| non-IBM premises | |
| terminals | 143 |
| "confined" accounts on APL | 43 |
| non-regular or non-IBM employees | 32, 137 |
| "confined" accounts on APL | 43 |
| Open House or Family Day | 43 |
| within a computing installation | 156 |
| non-trivial passwords | 31 |
| notification of supplier of services | |
| password compromises | 152 |
| notification of user management | |
| cancellation of services | 154 |
| NUMBER IN USE message | 25 |
| NUMBER LOCKED OUT message | 26, 30 |
| NUMBER NOT IN SYSTEM message | 24, 30 |
| numbers | |
| in restaurants | 112 |
| random | 109 |
object modules 149
objects 37, 185
observation
visual
within a computing installation 156
obvious passwords
data access 140
encryption keys 141
sign-on 139
ODTS 179, 180
office systems 128
offline storage
workspaces 37, 185
off-site storage of data 11
OLD PASSWORD INCORRECT message 28
one-liners 71
Opel, John R. 187
Open House 43
operating manuals 171
operational direction
extended term 175
short term 172
options
management 128, 171
Organizational Unit Security responsibilities 166
organization charts 171
output
classifying 135
partial unusable 158
physical access controls 158
outside access
to IBM systems 32, 142, 160
user requirements 153
overrun copies 158
overwriting files
preventing accidental overwriting 97
to dispose of residual information 83, 143
owner of data
definition 133
application design 148-149
assigning 132
genreal requirements 147
required functions in new applications 149-150
responsibilities 147, 150
basic 133
restricting access to 149, 157
owner of output 151
pages of output
   classifying 135, 170, 171
   control elements 139
   required for new applications 150
   restricted utilities 139

paper files
   See written procedures

paper tapes 169, 171

partial unusable output 158

)PASSWORD command 28

PASSWORD EXPIRED message 26, 30

PASSWORD function 33

password-change date
   at sign-on time 27

password datasets 138

passwords 39
   changing 28, 31
      (chart) 30
   concealing 21, 34
   generation and control 139

information access (data) 33, 140, 147
   definition 140
   blotting 34, 139, 149, 157
   classification 152
   compromise 152
   disclosure 152
   random generation 33
   recording 152
   sharing of 140
   symbolic parameters in indirect commands 90
   terminal display 152
   terminal entry 152
   user requirements 152

non-trivial 31

owners of 151

symbolic parameters in indirect commands 90

use of 28, 31

verification (sign-on) 33, 139
   definition 139
   blotting 21, 34, 139, 149, 157
   changing 30, 157
      (chart) 30
   classification 152
   controlling application access 149, 157
   deactivation 16, 30, 152, 157
   display 152
   entry 152
   for job initiation 157
length 140
random generation 33
recording 152
rules for creating 139
sharing of 25, 139, 152
supplier of services requirements 155
workspace 40
Payroll 137
PA2 key
trapping 62
trapping too much 69
use of 8, 49
during sign-on 22
PC See personal computer
periodic reviews
authorization records 155
by data owners 148
of data access 147
by management 138, 156
non-IBM terminals 142
self assessment and planning 144
inventories of portable storage media 159
of access authorization 16
for data access 147
for system sign-on 16, 30, 157
of billing 14
of compliance plans 144
of file classifications 82
of files that you own 73
of news items 5
of risk acceptances 144
of risk assessment and risk acceptance 148
of system utilization 14
of who you are paying for 14
Registered IBM Confidential (RIC) 178, 180
permanent datasets
creating 77, 79
personal computer 128, 133, 147, 155, 158
control of portable storage media 158
personal responsibility
assigning 159, 174, 177, 180, 181
personnel actions 146
personnel controls 156
computing installations 155
personal computers 147
small systems and input/output devices 158
supplier of services 154
personnel information 172, 173
Personnel Profile 170
PF1 key
use of
during sign-on 22
phone books 171
phone calls
Index

IBM Confidential (IC) 174
IBM Confidential-Restricted (ICR) 176
Registered IBM Confidential (RIC) 180
phone closets
  restricted access 156
phone numbers
  computer access
    classifying 158
    don't post 153
    protecting 142
    protecting 171
photographs 169
physical access controls 156
  computing installations 155
  personal computers 147
  small systems and input/output devices 158
  supplier of services 154
physical controls 145
Poe, Edgar Allan 12
portable storage media 135, 145, 170, 171
definition 158
  classifying 158
  control of 158
  inventories 147, 159
  marking 158
  residual information 143, 150, 158
powering off terminals
  user responsibilities 153
power supply areas
  near computing installations 155
practices
  documentation of
    owner's responsibilities 148
    supplier of services iii, 155
predictable passwords
  data access 140
  encryption keys 141
  sign-on 139
preliminary copies
  RIC 178
preprinting of classifications 170
prevention
  effective 146
  of repetitive attempts 149
printing precision (CPP) 46
  localizing 45
printouts
  classifying 135, 170, 171
  control elements 139
  required for new applications 150
  restricted utilities 139
privileged users
  password change requirements 140
problem number
definition 184
procedural requirements
responsibilities 144-147
production code 138, 139
products
compatibility with 141
security problems with 146
unannounced 172, 178
code names 171
use and modification 137, 164, 165
programmer
System Support
password change requirements 140
programmers
within a computing installation 156
programming
style 71
programs 37, 185
list 148
sensitive
access controls 157
auditability 146
controls 138
on portable storage media 159
owner's responsibilities 148
requirements 137-139
risk assessment and risk acceptance 145
source code 149
prompting for user keyboard input 55, 66
"Property of IBM" labels 159
protection
of information assets 133
protection of "*" for locking functions 52, 53
prototypes 171
pseudo-passwords
symbolic parameters in indirect commands 90
dpalette-random numbers 109
Public Library 39
punched cards 169, 171
Punxsutawney Phil 120
- input 66
  - alternatives to 66
- warnings against 55
- AT (accounting information) 47
  - example of use 6
- CR (canonical representation) 53
- CT (comparison tolerance) 45
  - localizing 45
- quad divide "%" 66
- EA (execute alternate) 67
  - See also dyadic execute 66
- EX (expunge) 49, 51
  - definition 184
- EX ("fix"—function establishment) 56
  - dyadic form 57
- IO (index origin) 110
  - localizing 45
- LC (line counter) 47
- LX (latent expression) 47, 48
  - definition 184
  - example of use 7
- WC (name classification) 71
- PP (printing precision) 46
  - localizing 45
- RL (random link) 47, 109
  - example of use 33, 113, 116
  - localizing 113
- SVC (shared variable control) 119
  - example of use 36, 58
- SVE (shared variable event) 119
  - example of use 122
- SVO (shared variable offer) 119
  - example of use 36, 58, 118, 122
- SVQ (shared variable query) 119
  - example of use 122
- SVR (shared variable retraction) 119
- SSV (shared variable state) 119
  - example of use 122
- TS (time stamp) 47
  - example of use 111
- input 55, 66
  - with execute 66

*Quote Quad* magazine 70, 117
RACF 137, 140, 147, 149, 157
  definition 185
RACF functions 125
RACF facility 125
random link (RDL) 47, 109
  example of use 33, 113, 116
  localizing 113
randomly-selected passwords
  PASSWORD function 33
  encryption keys 141
  sign-on 139
random numbers 109
“The Raven” 12
read access
  Confidential data 149, 150
  controlling 136
  repetitive attempts 149
  restricting 138, 147
  supplier of services responsibilities 157
receiving stations
  messages 143
reclassification of data
  supplier of services 154
recordings
  sound 169
record-level access control 149
records
  See also written procedures
    definition 169
recovery
  See dyadic execute
references 187
Corporate Security 161
Registered IBM Confidential (RIC)
  definition 178
  audits 178, 180
  copying 179, 180
  disclosure outside of IBM 181
  generating 178
  mailing 179
  reports of access 146, 157
  requirements for 178, 180
  risk assessment and risk acceptance 145
  telephone calls 180
  transmitting 179
  travel with 181
Registered Records Control Centers (RRCC’s) 179, 180
Reimbursement Accounting 137
remote computing 128
removal of classified information 156, 159
repetitive attempts to access data 149
reporting security exposures 132, 134, 144
owner's responsibilities 147
risk assessment and risk acceptance 144
unauthorized personnel 155
reports
classifying 135, 170, 171
control elements 139
required for new applications 150
restricted utilities 139
of access
Registered IBM Confidential (RIC) 146, 157
of encryption key changes 146
reproduction
IBM Confidential-Restricted (ICR) 176
Internal Use Only (IUO) 171
of software controls 138
Registered IBM Confidential (RIC) 179, 180
RESEND message 23
reserved datasets 85
definition 85, 185
inner workings (for programmers) 95
residual information
definition 143, 150
disposal of 143, 150
on portable storage media 158
Resource Access Control Facility
See RACF
responsibilities 132, 147
assigning 132, 158
supplier of services 154
basic 132-134
management 132
owner of data 133
supplier of services 134
user of services 134
bi-annual audits 16
classification of information 169
Corporate IS&A Staff 163
Corporate Security staff 162-166
Corporate Security Staff 162
Director of IS&A 162
Director of IS&A Staff 163
Director of Security 162
Director of Security Staff 162
IAS executive/manager 164
Information Asset Security 164
information classification and control 135-136, 169
management 132
Organizational Unit Security 166
owner of data 133, 147-150
periodic mailings 13
procedural requirements 144-147
senior security executive 166
specific requirements 137-143
Staff 162
supplier of services 134, 154-160
user of services 134, 151-153
restaurants
mathematics of 112
RESTRICTED COMMAND message 85, 104
restricted area
access logs
auditability requirements 146
Restricted (ICR)
definition 175
deletion of data 83
disclosure outside of IBM 177
overwriting prior to deletion 83
requirements for 175
terminal encryption 116
restricted libraries 42
restricted utilities
definition 138
access controls 157
auditability 146
controls 138
on portable storage media 159
owner's responsibilities 148
requirements 137-139
risk assessment and risk acceptance 145
restricting access
don-IBM users
“confined” accounts on APL 43
Open House or Family Day 43
to owner of data 149, 157
restrictions
documentation of
owner's responsibilities 148
supplier of services iii, 155
revalidation
of access authorization
for data access 147
for system sign-on 16, 157
of compliance plans 144
of risk acceptances 144
of risk assessment and risk acceptance 148
reviews
authorization records 155
by data owners 148
of data access 147
by management 138, 156
non-IBM terminals 142
self assessment and planning 144
inventories of portable storage media 159
of access authorization 16
for data access 147
for system sign-on 16, 30, 157
of billing 14
of compliance plans 144
of file classifications 82
of files that you own 73
of news items 5
of risk acceptances 144
of risk assessment and risk acceptance 148
of stable applications 148
of system utilization 14
of who you are paying for 14
Registered IBM Confidential (RIC) 178, 180
RIC (classification)
See Registered IBM Confidential
right-paren “)” 34
definition 185
risk assessment and risk acceptance
definition 144
annual requirement 148
Confidential (IC) 145
Confidential-Restricted (ICR) 145
IBM Confidential (IC) 145
IBM Confidential-Restricted (ICR) 145
Internal Use Only (IUO) 145
non-IBM terminals 143
owner’s responsibilities 133, 147
procedural requirements 144
Registered IBM Confidential (RIC) 145
revalidation 144
supplier of services 154
roll function (?n) 110
rough drafts
RIC 178
routine entry
into a computing installation 155, 156
RPQ’s
use of 137
RRCC
Registered Records Control Centers 179, 180
rules
documentation of
owner’s responsibilities 148
supplier of services iii, 155
safeguards
  physical and procedural 154
  safety 142, 143
salary information 172, 173
satellite transmission 116, 141, 176, 179
satisfactory compliance
definition 145
  causing unacceptable business impact 154
demonstrating 146
  internal applications 148
monitoring
delegation of authority 147
  non-IBM terminals 153
  owner's requirements 148
  supplier of services 154
  user's requirements 151
stable applications
  review 148
)SAVE command
  with a password 39
schedule
  system operation 10
  "scramble" (account number) 89
scrambling of data 113
secondary storage
  workspaces 37, 185
second (current time) 47
secured libraries 42
security 39
definition 1
  excuses 2
10 SECURITY (workspace) 32, 33, 116
security check-list 9
security exposures
  identifying 144
  risk assessment and risk acceptance 144
  in program products 145
  reporting 132, 134
  owner's responsibilities 147
  unauthorized personnel 155
Security Instruction #104A
  See "Corporate Security Instruction #104A"
security references 161
security requirements
  minimum 128
security responsibilities
  staff 162-166
self assessment and planning 128, 132
definition 144
  annual requirement 148
annual reviews 144
  documenting 146
  non-IBM terminals 143
  owner's responsibilities 133, 147
  procedural requirements 144
semaphores 97-100
  shared 100
sending stations
  messages 143
senior security executive responsibilities 166
sensitive program
  definition 137
sensitive programs
  access controls 157
  auditability 146
  controls 138
  on portable storage media 159
  owner's responsibilities 148
  requirements 137-139
  risk assessment and risk acceptance 145
separator pages
  classifying 135, 170
Series/1 155
server machine 120-123
  definition 120
10 SETBOMB (workspace) 61
shared systems
  controlling access to data 85, 147, 157
  alternatives 147
shared variable control (DSVC) 119
  example of use 36, 58
shared variable event (DSVE) 119
  example of use 122
shared variable offer (DSVO) 119
  example of use 36, 58, 118, 122
shared variable query (DSVQ) 119
  example of use 122
shared variable retraction (DSVR) 119
shared variables 118
  table 119
shared variable state (DSVS) 119
  example of use 122
sharing of passwords
  data access 140
  sign-on 25, 139
  user requirements 152
shooting the messenger 2
)SI command 37, 55, 185
signing in 155
signing off
  terminals
  user responsibilities 153
signing on 21, 34
  format of input 24
  format of output 27
logging of attempts 27
one-minute time-out 27
telephone access 19
See also information inside front cover
sign-on number 24, 47
sign-on passwords 28
See also passwords, verification
simultaneous updates
handling 97-100
single-person access 138
)SINL command 55
1 SMF (workspace) 101-107
Smith, Howard J., Jr. 187
software
classifying 136
software controls
misuse or avoidance 138, 146
through repetitive attempts 149
software extensions
of IBM products 137
SORTBYCLASS function 81
sound recordings 169
source code changes
of software 138
source programs 149
specification
to a supplied name 65
specifications
for software 138
stable applications
exemptions 148
Staff responsibilities 162
stand-alone functions
recommendation against 54
stand-alone systems 128, 133, 147, 155, 158
control of portable storage media 158
stapling
obscurity classification 170
state indicator 37, 185
state indicator commands 55
statement numbers 47
STATUS function 81
storage
portable storage media 159
storage of datasets 73
style
programming
See programming style
sub-functions 54
supervision
management 137
during testing 149
SUPERZAP 138
supplier of services
definition 134
general requirements 154
responsibilities 154, 160
  basic 134
supporting facilities 155
telecommunication
  restricted access 156
surveillance
  visual 155
    within a computing installation 156
suspension of functions
  preventing 52-54
  restarting 48
symbolic parameters in indirect commands 86
  pseudo-passwords 90
  reserved names 87
^SYS prefix in symbolic parameters 87, 88
System/7 155
system and information access controls 157
system command
  definition 185
system commands
  effects of 37
    effects of (picture) 38
    for libraries 39, 184
system control centers
  near computing installations 155
system operating schedule 10
System Support
  password change requirements 140
system variables 45-50
  localizing 45-46, 113

table of contents iv, 130, 168
tape/disk libraries
  near computing installations 155
tapes
  control of 158
  magnetic 163, 171
  paper 169, 171
  storage of datasets 77, 83
technical guidance
  supplier of services 160
telecommunication facilities
  restricted access 156
telegrams
IBM Confidential (IC) 173
IBM Confidential-Restricted (ICR) 176
Registered IBM Confidential (RIC) 179

telephone access
for signing on 19
See also information inside front cover
bad connection 23
telephone calls
IBM Confidential (IC) 174
IBM Confidential-Restricted (ICR) 176
Registered IBM Confidential (RIC) 180
telephone closets
restricted access 156
telephone directories 171
telephone numbers
computer access
classifying 158
don't post 153
protecting 142
protecting 171
temporary versus permanent datasets 77
terminal-ID 46
terminal rooms
near computing installations 155
terminals 142
at home 142
supplier of services responsibilities 160
user requirements 153
classifying displays 135
control elements 139
required for new applications 150
restricted utilities 139
encryption for IBM Confidential-Restricted (ICR) 116
“management-approved purposes” notice 153
not under IBM control 32, 142
supplier of services responsibilities 160
user requirements 153
on non-IBM premises 32, 142
user requirements 153
positioning 153
signing off
user responsibilities 153
supplier of services responsibilities 160
unattended
user responsibilities 32, 153
unauthorized viewing 32, 153
user of services 153
user requirements 153
termination
of business needs 139
data access passwords 140
encryption keys 142
for data access 147
for system sign-on 16, 157
notification of owners 151
notification of supplier of services 151
sign-on passwords 140
testing
of applications
Confidential data 149
owner's requirements 148
text and messages 143
text processing 128
text protection 128
tie-line network 19
See also information inside front cover
time
compute
definition 184
compute time 47
connect
definition 184
connect time 47
keying time 47
"timely, effective response" 132, 134
losses and improprieties 146
supplier of services 154
to data access attempts 158
time-out
at sign-on time 27
time period
for changing passwords
(chart) 30
data access 140
privileged users 140
sign-on 140
for inventories of data 159
time-sharing systems
controlling access to data 147, 157
alternatives 147
time stamp (TS) 47
example of use 111
toothpaste 62
top sheets
classifying 135, 170
RIC 178
traceability
of user identification codes 157
transferring phone calls 20
transferring portable storage media 159
transmission of data
cryptography 116, 141
electronic 128, 143
IBM Confidential (IC) 173
IBM Confidential-Restricted (ICR) 116, 176
Registered IBM Confidential (RIC) 179
interception 141
via microwave 116, 141, 176, 179
via satellite 116, 141, 176, 179
via wire only 116, 141, 176, 179
within an IBM facility 116, 141, 176, 179
transmittal forms 159, 170
transmitting
IBM Confidential (IC) 173
IBM Confidential-Restricted (ICR) 116, 176
Internal Use Only (IUO) 177
Registered IBM Confidential (RIC) 179
trapping errors
See error side-tracking
traps 67, 69, 71
travel
with IBM Confidential (IC) 174
with IBM Confidential-Restricted (ICR) 177
with Internal Use Only (IUO) 177
with Registered IBM Confidential (RIC) 181
trivial passwords 31
data access 140
encryption keys 141
sign-on 139
TSIO
definition 185
TSIO “scramble” 89
TSIO datasets
command 85
definition 85
inner workings (for programmers) 95
symbolic parameters 86
how to classify them 79
keeping track of them 73
reserved 85
definition 85
inner workings (for programmers) 95
TSO UADS dataset 137
Tuttle, Joey K. 187

UADS dataset 137
unacceptable business impact
risk assessment and risk acceptance 154
unannounced products 172, 178
code names 171
unattended terminals
user responsibilities 32, 153
unauthorized use
of software controls 138
unauthorized viewing
terminals 153
underground storage of data 11
"unscramble" (account number) 89
unusable output 158
update authority
approval for 138
Confidential data 149
restricting 138, 147
supplier of services responsibilities 157
USAGE function 81
use and modification of IBM products 137, 146, 164, 165
user-friendly 69
user exits
in IBM products 137
user number (APL sign-on number) 24, 47
user of services
definition 134
access controls 152
as owners 151
computing installations 155
general requirements 151
responsibilities 151, 153
basic 134
terminals 153
utilities
restricted
access controls 157
auditability 146
controls 138
on portable storage media 159
owner's responsibilities 148
requirements 137-139
risk assessment and risk acceptance 145

value of assets
judging 133
Van Der Meulen, Mike 187
variables 37, 185
definition 185
erasing 51
global 54
localizing 113
shared 118
table 119
vendor services 128
“confined” accounts on APL 43
violation detection 137, 138
supplier of services 154
visible identification
badges 156
visitors
access controls 156
around personal computers 147
computing installations 155
small systems and input/output devices 158
supplier of services 154
within a computing installation 156
visual surveillance 155
within a computing installation 156
voice protection 128

waste
classified 142
“When HARLIE Was One” (novel) 29
wire rooms
restricted access 156
word processing 128, 148
workmen
within a computing installation 156
workspace 37, 185
definition 185
CONTINUE 41, 42
created automatically by the system 40
load-only 42
lock
See passwords, information access
workspace 1 AIDS
ALLOCATE function 79
for classifying files 79
IC function 85, 92
inner workings (for programmers) 96
INITIALIZE function 80
RACF functions 125
workspace 1 CATALOG
for locating Public Library workspaces 39
workspace 10 DOC104 2, 127, 167
workspace 1 FILELIB 73
CLASSIFY function 81
DELETE function 83
for classifying files 79
for identifying files 73
workspace 10 LOCALIZE
  for localizing functions 59
WS LOCKED message 40
workspace 1 NEWS 5
  security of 6
  sending your own messages 6
workspace 10 SECURITY 32, 33, 116
workspace 10 SETBOMB 61
workspace 1 SMP 101-107
work stations
  location of 155
written procedures
  requirements 138, 145, 146
    risk assessment and risk acceptance 146
    self assessment and planning 146
RIC 180
WRONG PASSWORD message (non-existent) 24
  See also OLD PASSWORD INCORRECT message
ws
  definition 185
WS FULL message
  definition 186
  cause and recovery 186

XIM function
  for data unscrambling 113
Controlling the Security of Applications on Kingston APLSV

—The APL Jot*Dot Times—

Y

year (current date) 47

Z

Zink, Achim 123
Ziskind, Marty 123
...Are we done?....

...Now, just what should I type to make my stuff "secure"??

The End
NOTICE

regarding phone calls to Kingston, using the IBM internal tie-lines

Effective 01 June 1985, the tie-line phone access for the entire Kingston IBM site will change from "373-" to "695-".

- If you call The APL Hotline before 01 June 1985, use T/L 373-1234.
- If you call The APL Hotline after 01 June 1985, use T/L 695-1234.

(Outside phone access will not be affected)
We are interested in hearing your comments... and perhaps, so are others. We would like to publish some of the comments in future issues. If you don’t want your comments to be published (or if you don’t mind them being published but don’t want your name to be attached), say so... we’ll honor your request. In lieu of such a note, any of the comments will be considered to be fair game for publication.

Time will undoubtedly prevent us from personally responding to very many of the comments that are sent in.

1. What would you like to see in our future newsletters? [Also, of course, what should we omit?] What did you like in this one? What did you hate?

2. What would you most like to see the APL Support group work on; what projects would be the most useful to you?

3. Suggestions or comments on anything else:

--- Thank you for your time! ---

Name __________________________ IBM Address __________________________ T/L __________________________
Designing "Bullet-Proof" Code

EVERYTHING you ever wanted to know about APL Security
...plus 234 additional bonus pages!