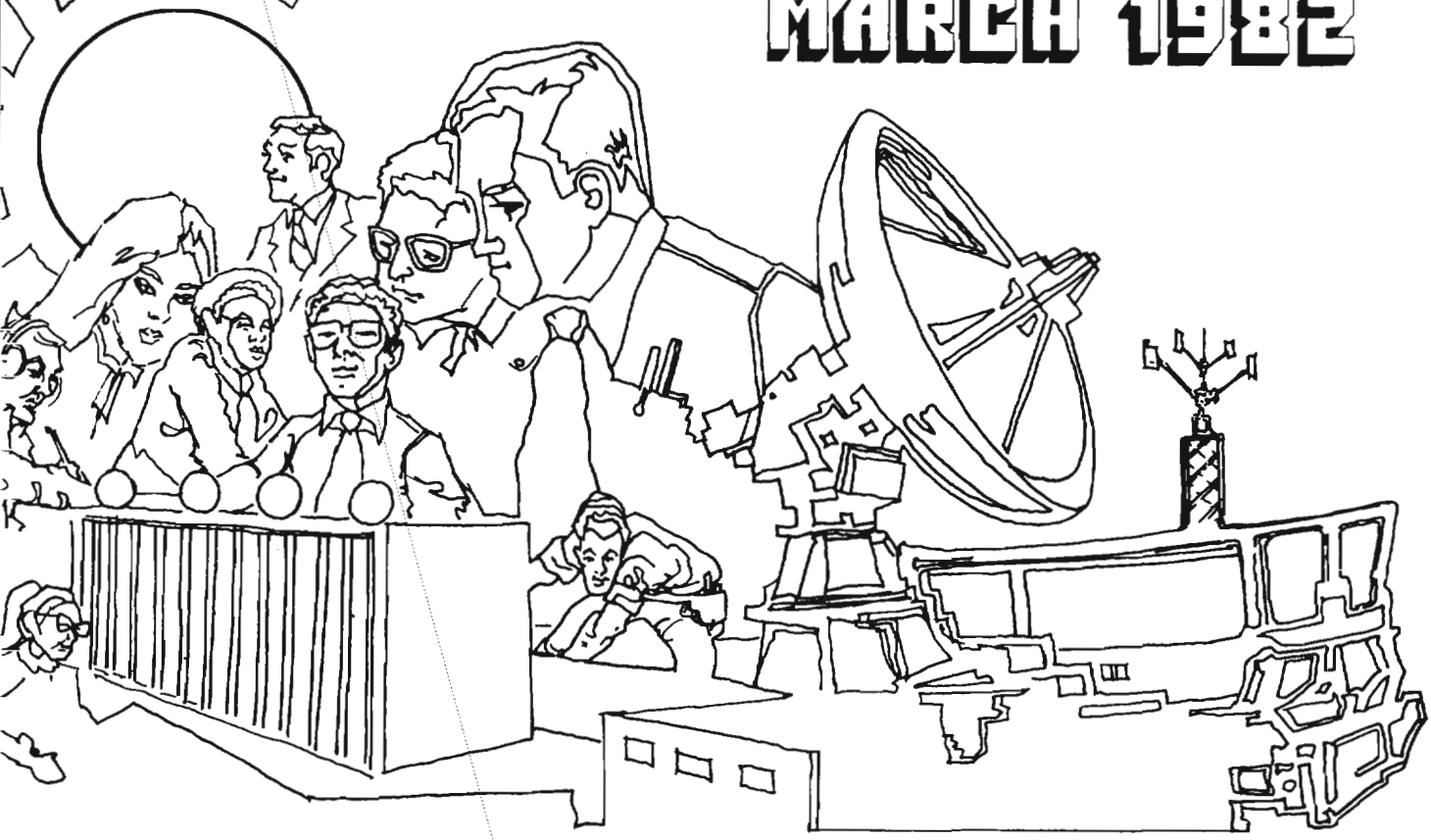
~~TOP SECRET~~

NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND

CRYPTOLOG

MARCH 1982



~~THIS DOCUMENT CONTAINS CODEWORD MATERIAL~~

~~TOP SECRET~~

CRYPTOLOG

Published by P1, Techniques and Standards,
for the Personnel of Operations

VOL. IX, No. 3 MARCH 1982

For individual (or organizational) subscriptions send name and organization

to: CRYPTOLOG, P1
or call [redacted] 3369s

To submit articles or letters via PLATFORM mail, send to

cryptolg at barlcØ5
(note: no 'O' in 'log')

Editorial

One of the interesting aspects of editing a publication is the challenge of finding out what the readers think. One of my predecessors has said that, on a scale of one to ten, a single non-angry letter is an eight, whereas an angry letter is a two (or maybe a one). It is a fact, that angry readers are more likely to write (or throw rocks through the window with a message attached -- should that be counted as two messages?) than are the satisfied (or edified or entertained or whatever) readers. How then is one to know if there <u>are</u> any non-angry readers?

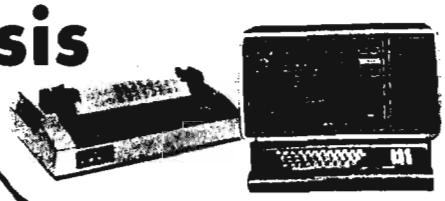
All of this makes me wonder if we should deliberately put in something to angrify the readers, every so often. We could declare a "Letter Month" and see what turns people off. It probably isn't a good idea to do it deliberately; there are too many "pattern finder" people around here, and as soon as they tumbled to the pattern, they would lose interest, their adrenalin would decline, and so would the volume of mail.

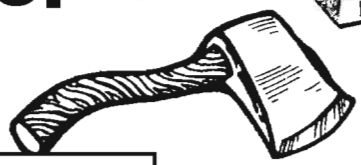
The system we have now might do us just as well. It seems to be working pretty efficiently. In a recent issue, the wrong version of an article was sent to the printer; one angry response came to us in technicolor. It is being framed. If the sender had included his/her name, we might have asked him/her to help us as a proofreader. We can probably make enough of these mistakes without trying to schedule them formally.

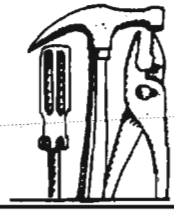
Which brings us to a related point: the errors in the magazine as you readers see it are almost always mine, rather than the author's; I am the typist/checker, so if something offends you, shout at me, not the author.

P.L. 86-36

The PERSONAL Computer: A *Current* Cryptanalysis Support Tool (u)



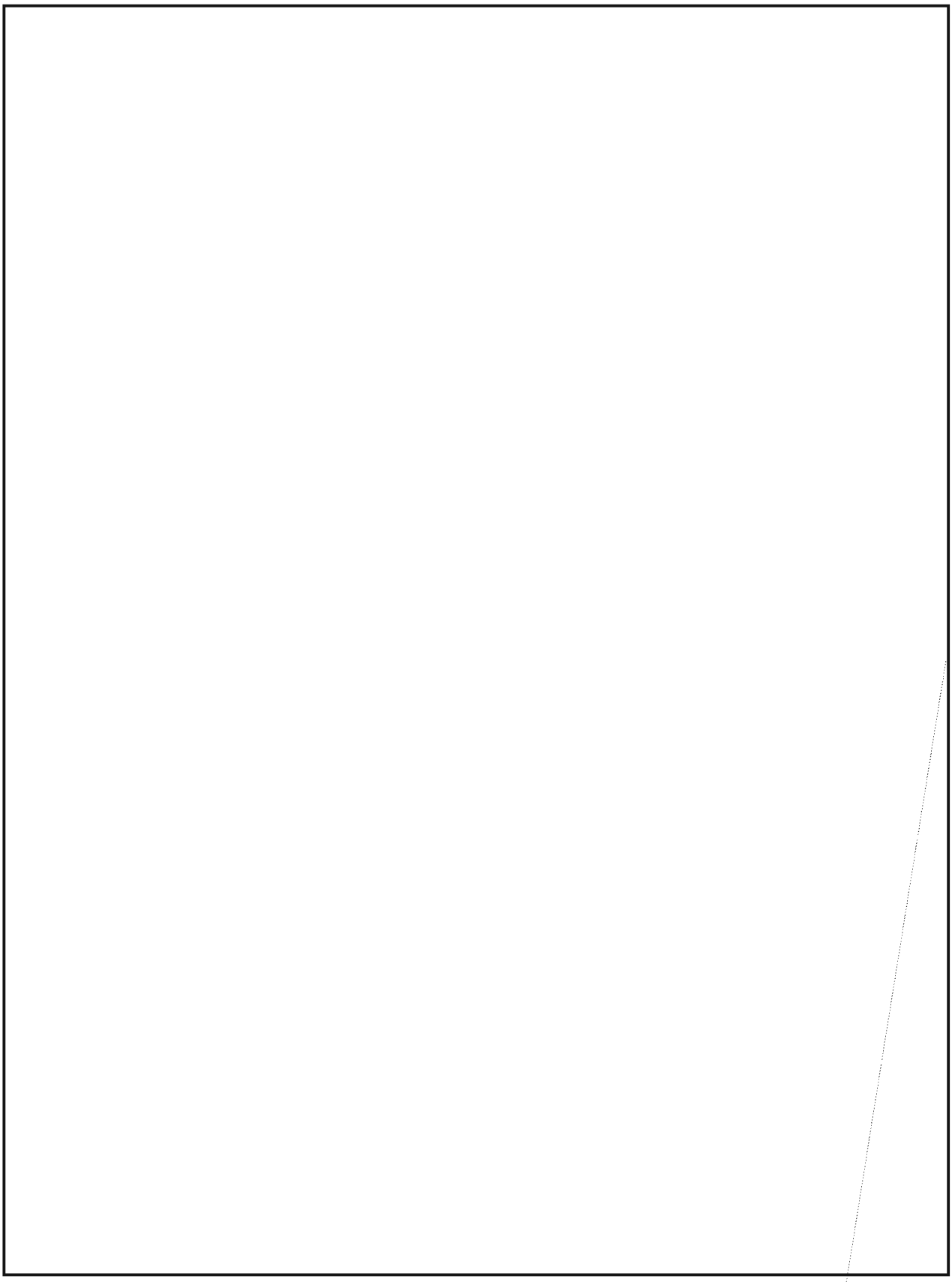




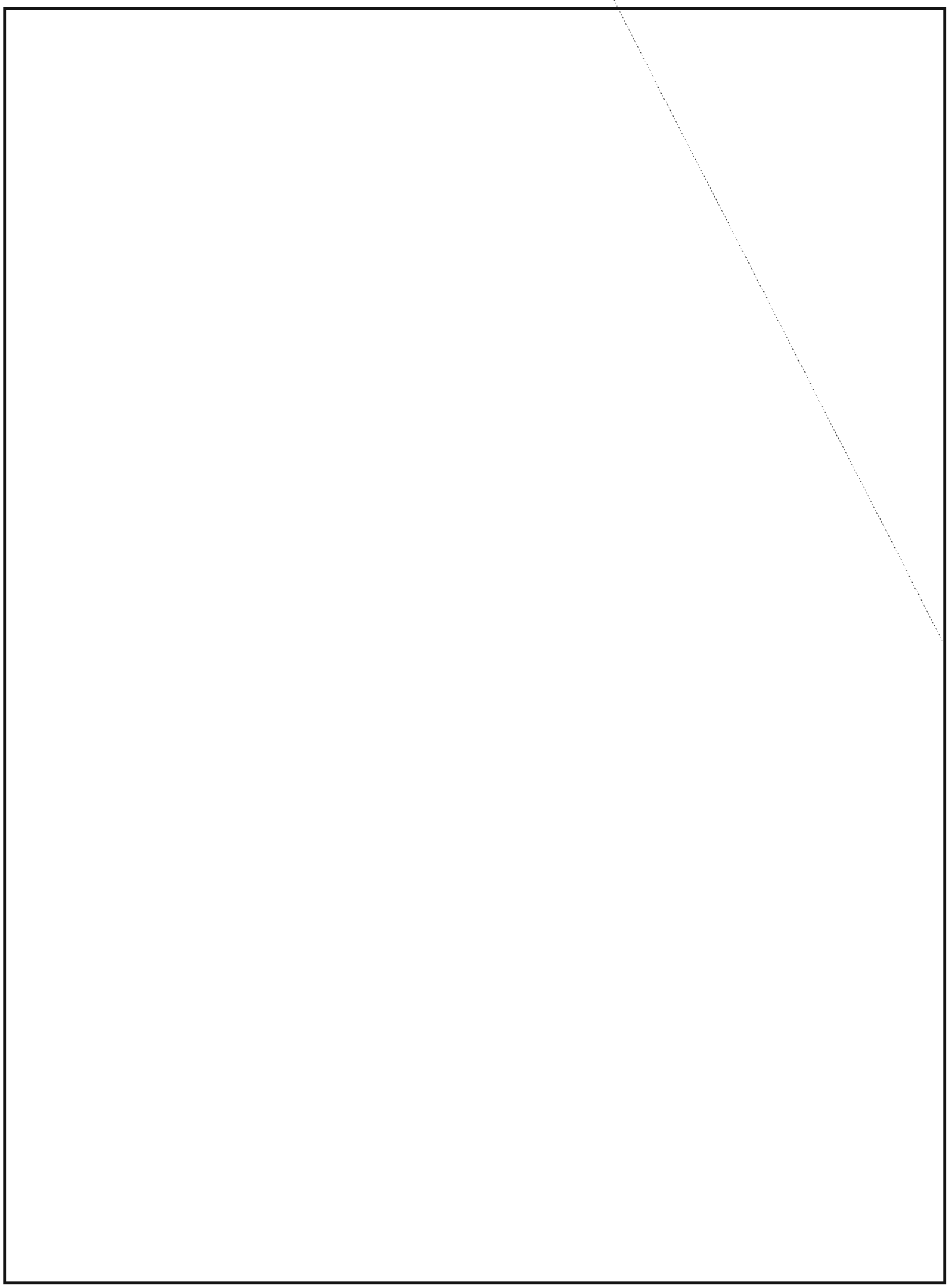
by [redacted] B53

P.L. 86-36

EO 1.4.(c)
P.L. 86-36



EO 1.4.(c)
P.L. 86-36



A Historian Looks At SIGINT(U)

by Vera R. Filby, E41

Eighteen senior NSA executives, along with a few representatives from the National Cryptologic School and the Director's staff, attended a seminar in the Director's Conference Room on 18 February to hear Ronald Lewin, author of ULTRA Goes to War and the recently published (February 1982) The American MAGIC: Codes, Ciphers, and the Defeat of Japan, speak on "SIGINT and Decision Making in War." The occasion was the second in a series of seminars for senior executives initiated and arranged by E6, the Cryptologic Management Department of the National Cryptologic School. At the first seminar, former Senator Birch Bayh spoke to a similarly small, selected group.

In his introductory talk before questions and discussion, Mr. Lewin presented conclusions to which his researches in World War II signals intelligence has led him. Among these were

- that the major decisions of governments were based not on intelligence but on political considerations;
- that ULTRA's most valuable contribution was the day-in, day-out accumulation of order-of-battle information; and
- that constant familiarization of intelligence staffs and some commanders with the flow of signals intelligence led to a climate of understanding;
- so that a solid base of data was the greatest contribution of signals intelligence to decision making.

Of the commanders who understood and appreciated it, he mentioned particularly Alexander, who every day "soaked himself" in the production of his "ULTRA room," and Nimitz, who understood the value of SIGINT from the beginning. Mr. Lewin stated and repeated that no praise is high enough for Nimitz as a commander.

After a coffee break, questions led to a variety of answers and issues. Among Mr. Lewin's responses were these:

- intelligence must work in action because that is what it exists to do, and
- intelligence has failed because it did not conform to the pictures the leaders had in their minds, and they were therefore unwilling to believe it. He later made the point that the value of intelligence is not in having floods of it but in having it right.

A question on Anglo-American cooperation led to Mr. Lewin's evaluation that after its tentative start in 1941 ("careful prowling around," he called it), the relationship by 1944 had become very, very close. Each country had something to give the other. Both had systems of belief in common. SIGINT

cooperation varied in different theaters, being least effective in the Asian and Pacific areas; but in the Atlantic and Europe it was just about perfect -- "a many-sided miracle, an example of history."

There were questions on his chapter "The Stab in the Back" in The American MAGIC. Mr. Lewin's thoughts on security and protection of signals intelligence were expressed on several levels. First of all, he said he wrote the chapter out of a sense of tribute to the dedication of cryptanalytic people. For Americans, one of the central problems was "walking a tightrope on the very highest wire." On the one side was the vital importance of protecting the information; on the other the danger of its annihilation by those who would ruthlessly expose it for unprincipled, political reasons without regard for the national welfare. He noted that the British early centralized the control of intelligence, that the Americans were less successful in resolving service rivalry -- that the Germans were riven with jealously protective rival services. The lapses of American security were of intimate concern to the British; it was their security too at stake. Even now, Mr. Lewin remarked, America has no viable way to manage the problem.

In response to questions about his evaluation of signals intelligence in war, Mr. Lewin stressed its use to the commander. (This was his theme in ULTRA Goes to War: the battle is the payoff.) In war, intelligence is useful only to the extent that it contributes to the central action, that it is present positively in the area of operations, that it meets the needs of commanders. And to this he added his conviction that commanders must understand and that they must have intelligence officers to help them understand. People believe what they want to believe. The problem is to get the truth to them.

Mr. Lewin is a military historian and biographer (Rommel, Montgomery, Churchill, Slim, Wavell), and his SIGINT perspective derives from information released to the public record since the mid-1970's. With his vast knowledge of World War II history, he has this been able to discern and examine the previously invisible presence of SIGINT in the history of the war. Listeners found his insights most interesting and perhaps a little disconcerting.

ATTENTION AUTHORS AND PUBLICITY CHAIRMEN

CRYPTOLOG is always looking for readable material. It is not a substitute for formal product or technical reports, but it does reach most of the people who are interested in the "technical underside" of our business, to borrow (and modify) a phrase. We are now publishing once a month and hope to maintain that schedule.

How can you get an article into CRYPTOLOG? Contact any one of the editors listed on the inside front cover. We don't need an abstract, and only one copy is necessary, as long as it is readable. If it is typed using a Mag Card, we can borrow your cards and use your original keystrokes. If it is typed into a UNIX system, send it to us, using the UNIX 'sndmsg' command:

- send it to

  cryptolg@barlc05

  (no O in 'log' and no spaces)

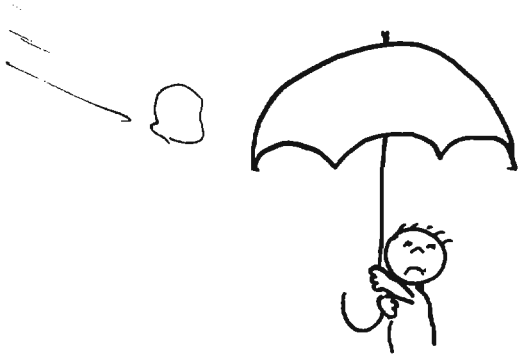
- after you have typed in 'Control D', type 'f' for file, and then name the file your article is in.

- WARNING: send only alphanumeric files this way. If your file contains any special characters, you will have to use ftp, or cftp. Give us a phone call, and we'll let you know what to do.

If your article is on some other machine, give [redacted] a call. We may still be able to use your keystrokes, especially if your computer is connected to PLATFORM. If it is not on any system, don't worry; we will type it.

If your organization has a publicity chairman, or program director, have him/her get in touch with us. We can get your news and announcements out to a large distribution list, and we would be happy to work with you to get the word out about what your group is doing.

METEORBURST Communications (U)



LOOKING AROUND with [redacted] P13

Meteorburst communications (MBC) systems for tactical traffic are being offered for sale by two U.S. manufacturers, ROCKWELL and TELCOM.[1] There has been some Soviet interest, in Soviet literature, in use of meteorburst communication for rapid delivery of a short message.[2] The Department of Energy (DOE) is developing an "emergency communication" (ECOM) network for its atomic energy facilities using a sophisticated meteorburst system that can operate through a nuclear war. Several large scale networks, viz. the SNOTEL net in the Rocky Mountains, and the AMBCS net in Alaska, have been operating for years under severe weather and propagation conditions, demonstrating the cheapness and reliability of this mode of communication.[3] Meteorburst has also been considered for the "strategic connectivity" problem, where low bandwidth links are use to assign and set up surviving wideband links. The fact that several competing companies are in the marketplace, and have been buying up competitors, implies that a market potential is seen. In addition to the obvious military market, the third world civil networks are a likely place to apply this low cost technology, because of the shortage of shortwave frequencies.[4]

Every day some ten billion tiny meteors, consisting of dust particles orbiting through the solar system, enter the earth's atmosphere and leave brief ionized trails which can reflect radio waves. Radio signals in the range 20 MHz to 150 MHz can be reradiated from the meteor trails, and if the trail geometry is correct for a given path, some of the energy will be scattered down to the receiver. The paths only last for a fraction of a second, so typical systems operate at high bit rate bursts of 2000 to 8000 bps, with a duty cycle (active path to the receiver) of 2 to 20 percent.[5]

Because of the short path duration, and random occurrences of useable paths, voice communication is impractical, but telegraph circuits work very well. While circuit control was done by relays and conventional teleprinters, the systems were not very useful, even though research and experiments have been carried out since the early 1950's. The introduction of microprocessors has changed the economics and operational attractiveness of meteorburst, by shielding the communication operator from the vagaries of the propagation. ARQ systems send little mini-packets of data at high bit rates, so that an average of 1 to 4 teleprinter links is maintained. Soviet tests showed quick delivery of short messages because very brief trails could be used.

The main features of the modern systems are: path lengths up to 1200 miles. Power between 10 watts and 1000 watts, depending on the grade of service and throughput wanted. Modulation may be FSK or PSK, up to 8PSK. Bit rates during bursts are typically from 2000 bps to 8000 bps (at 8PSK). The outstations in the SNOTEL net, with over 475 stations, cost from $7000 to $10,000 (depending on who is asked). The DOE ECOM net will be a full duplex grid with about 17 stations, and the specification calls for ECOM to pass a 900 character message across the U.S. between any two stations in 90 seconds. The software development to assure this, plus reconnectivity, is the problem. ECOM will use 41 and 48 MHz channels, but frequency ranges between 30 and 100 MHz provide practical service.

A small company in Texas did the basic engineering and hardware of the DOE SNOTEL net, with Western Union acting as prime contractor. ROCKWELL has bought up the SECODE company. TELCOM wrote the RPQ for the DOE ECOM net, for which Western Union is also prime contractor. The SNOTEL net was used to telemeter snowfall data from inaccessible

mountain slopes in the Rocky Mountains at a rate of a few reports per day, in response to polling from a central station. The equipment was robust, and operated from solar powered batteries at a few watts radiated during the brief transmissions. The antennas are smaller than an outdoor TV antenna, and this makes them suitable for tactical and remote area transmission.

Meteorburst might be useful for strategic connectivity within a country or a continental area, with relay hops of up to 1200 miles, to connect military and weapons bases and other vital facilities, because it is narrowband and very flexible. The unit costs can be reduced by competition and large scale manufacture. Frequency reuse can be fairly high, unlike HF, because the forward scatter mode make mutual interference very unlikely. It can also be used for theater area fallback, dedicated and reconnection circuits, without having to battle through the HF medium.

Unlike HF, meteorburst links can operate on a fixed frequency 24 hours a day. At dusk, as the earth sweeps away from the meteor showers, traffic capacity drops, then reaches a maximum at 0600 local time. The ability to use a single frequency on a link, and to reuse frequencies, facilitates the frequency management. Because interference is negligible, and the propagation and scattering is directional, international coordination or registration of frequencies is unnecessary. Therefore, dedicated domestic and military nets can be set up and run without formal notifications to ITU.

For transoceanic paths, the possibilities are surprising. On the Atlantic route, Scotland, Iceland, Greenland, Canada is obvious. Azores-Canary-Cape Verde-St. Peter St. Paul to Brazil is another route. Or a link can run from the Azores to Miquelon to Canada. It's also surprising how many routings there are via islands owned or governed by the U.S., U.K. or Commonwealth, and France, that allow hops across the South Atlantic, Indian Ocean, Antarctica and Pacific. Access to the Mid East and Persian Gulf areas is possible by 1200 mile hops from places where U.S. or friendly bases or diplomatic installations exist. The equipment is so small and cheap that it can be installed on small or large vessels, and left unattended on uninhabitable islands, deserts or mountains.

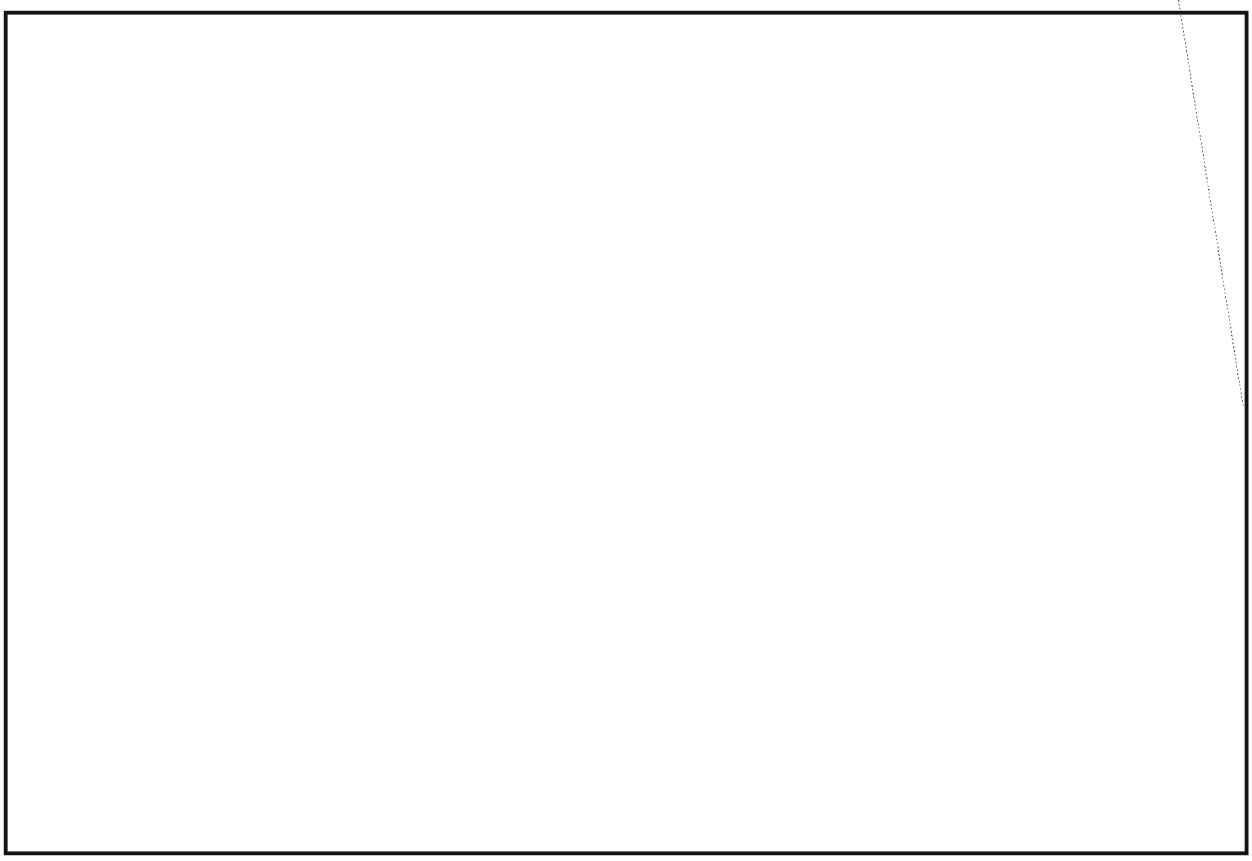
The attraction of meteorburst have been known for a long time, and several times the U.S. and NATO military departments nearly went ahead with projects, but in the early 1960's the attractions of wideband satellites overshadowed MBC. Now, because of the connectivity problem, to say nothing of the gross vulnerability of almost all military and civil telecommunications to many threats, MBC may look rather more attractive. The low cost of the terminal equipment, and the availability of working hardware and software from competing suppliers -- with the convenience of microprocessors for link and message handling -- might make this an attractive fallback option. There has been some Amateur interest in MBC, and the increased availability of Amateur digital radio equipment may stimulate further Amateur MBC. This cheap over the horizon medium should be very suitable for special forces and missions, and rapid deployments, because they can take their own dedicated equipment and their own frequencies, and be supported by a base hundreds of miles away which has satellite terminals to connect it into a global net.

Interference from TV broadcasting would discourage MBC in urban areas, but in remote areas or in war operations such TV interference is very unlikely.

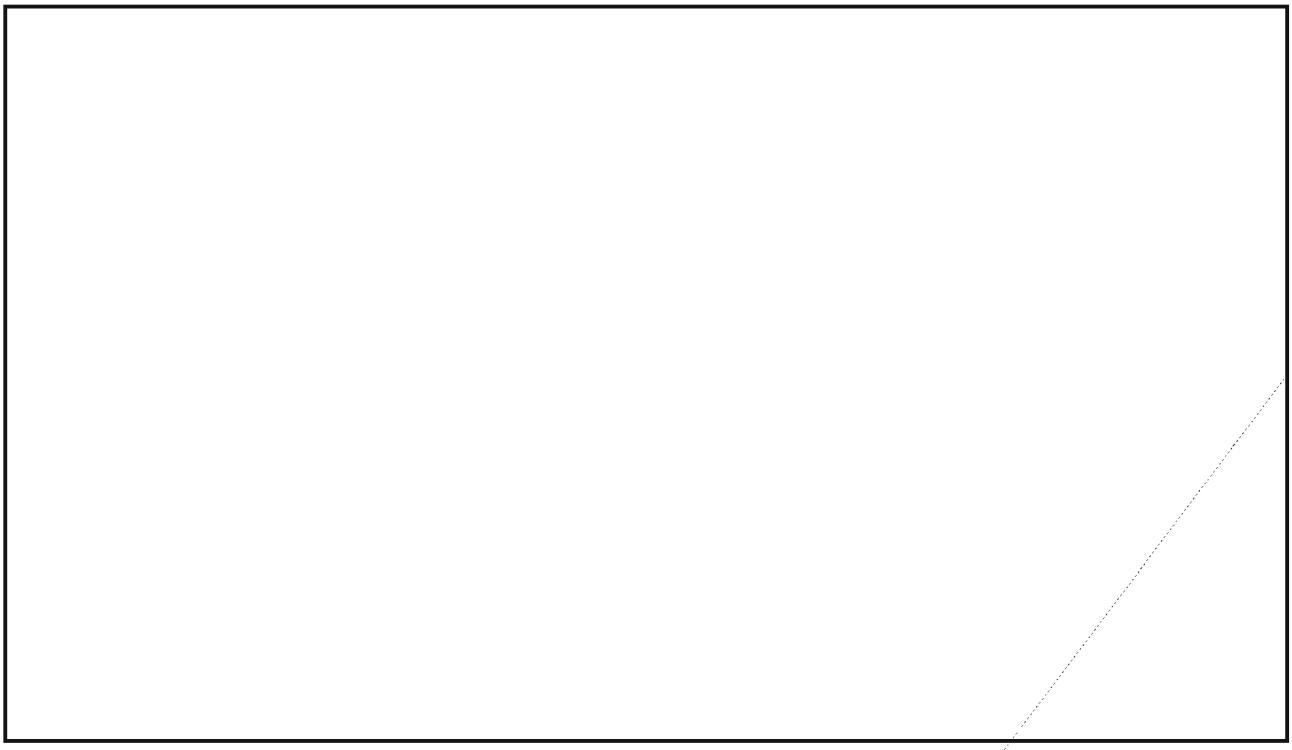
Summing up: cheap, flexible, hard to intercept, almost impossible to jam, low power, lightweight, fixed or variable frequencies, long range, independent of all other power or communications systems, with 50 to 300 bps average throughput, and high reuse of frequencies. How have military communicators managed without this?

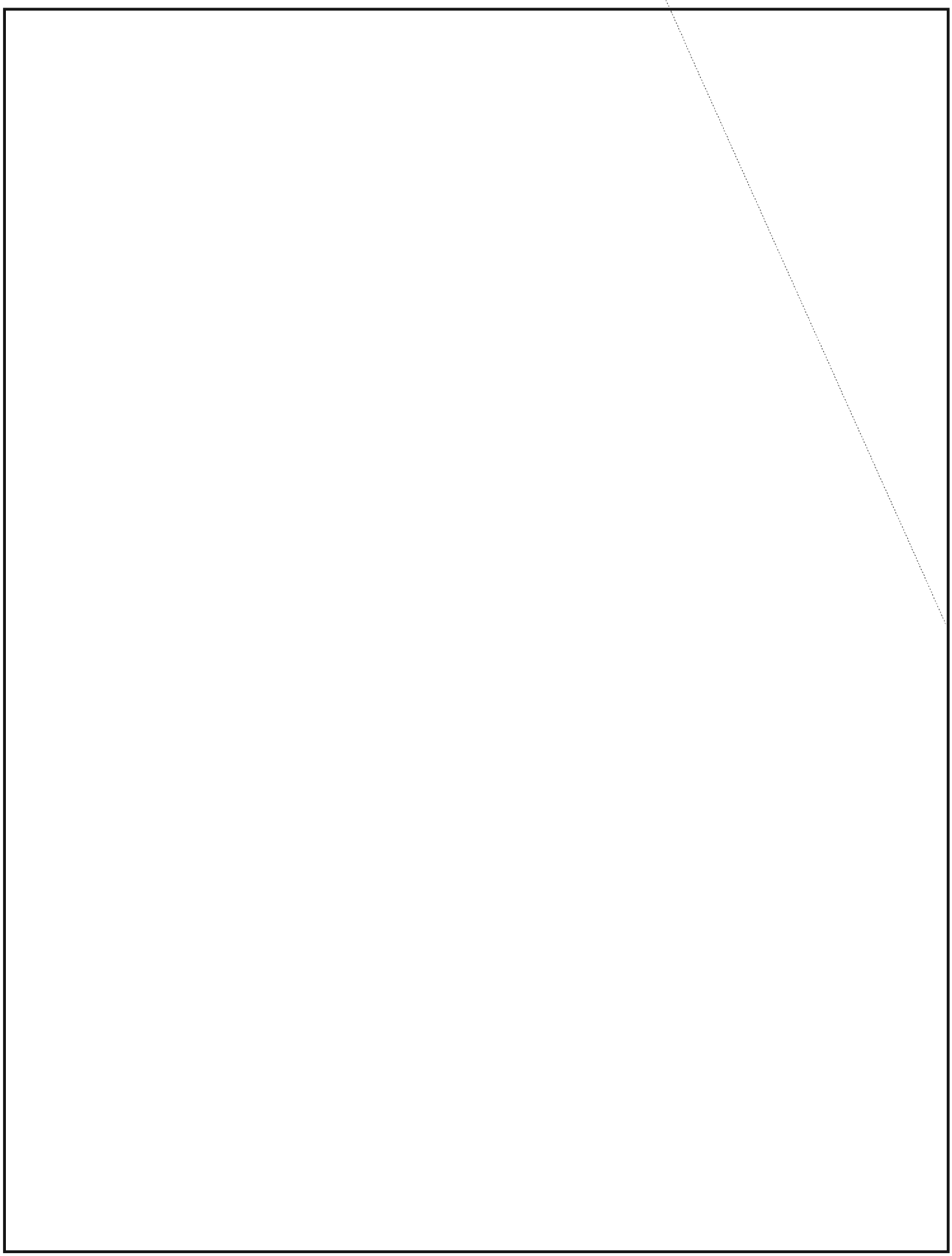
References:

1. AFCEA 81, ROCKWELL exhibit. Product information sheet "Meteorburst Communications". TELCOM advertisement in SIGNAL, 1981.
2. E.A.Demin et al, Test Results of Equipment for Data Transmission via Meteorburst Channels of Radio-Telegraph Communication Links, Telecommunications and Radio Engineering, 1977, No.1, Scripta Publications D.C. 1977.
3. Western Union data sheet. "Meteor Burst Communications" AFCEA 1979. Rockwell International data sheet, "Meteor Burst Communications", AFCEA 81.
4. J.A.Meyer, Digital Meteorburst Communications, INTELCOM 80 Proceedings.
5. D.W.Brown, The potential of meteorburst communication, Communications Systems and Random Process Theory, Ed. J. Skwirznski, Sijthoff & Noordhoff, Holland, 1978.



by [redacted] P16





A Brief Treatise on Five Laws of Telephonic Communication (U)

by William M. Nolte, T54

LAW #1: YOUR FIRST CALL OF THE DAY DETERMINES THE KIND OF DAY YOU'RE GOING TO HAVE.

This law is a derivative of the street vendor's belief that the day's success is determined by whether he makes a sale to his first customer. Cut the price, give away the merchandise if you must, but make that first sale!

The same principle applies to using the telephone: if the first call goes through to the intended recipient, you will have a favorable success rate on the rest of the day's calls. Unlike the vendor, who works in a "yes-no" situation (he either makes the call or he doesn't), the telephone caller works in a more complex situation. The call can be successfully completed of course, or it might result in a busy signal -- the rough equivalents of the "sale - no sale" conditions encountered by the vendor. The telephone involves. however, a series of other conditions, all indicating a lack of success, and each of them worse than the "no sale" or busy signal response.

- Subcondition A: "I'm sorry, he's on the other line. May I have him return your call?" Sure, but nobody stays on the phone for more than a few minutes around here (on official business, at least), and the mean time for a return based on such an assurance probably drags into days, leading to Corollary A:
  - ANY CALL RESULTING IN THE RESPONSE DESCRIBED IN SUBCONDITION A WILL EITHER BE RETURNED WITHIN TEN MINUTES OR WILL NOT BE RETURNED AT ALL.
- Subcondition B: "She's not at her desk." The truly significant aspect of this subcondition will be revealed to the caller with the next sentence. There are two possibilities:
  - "She's in a meeting with so-and-so, and I expect her to return at 1000." This person is clearly organized enough to leave a trail, and is therefore a good be to return your call. Action: leave a message to return your call.
  - "Gee, she was here just a minute ago, but I don't see her and I have no idea when she'll be back." Now you're in trouble. You have clearly called a floater, who could be in the bathroom, another office, the drug store, or fifty other places. It is unlikely that he or she is working. Action: say you will call back and take your chances. If you feel especially daring, leave a message to have your call returned. Do not however, postpone your own trips to the bathroom, cafeteria, or any of those other places to await the call. In fact, the worst way to get this person to return your call is to sit by the phone for three hours ignoring persistent signals from your stomach or bladder. The best way, on the other hand, is to leave your desk for thirty seconds to deliver a memo to someone in the next office. You guessed it, your call will be returned while you are gone.
- Subcondition C: "He's no longer at this extension. Let me see if we have a current number. This leads to:

LAW #2: NEVER PROCEED PAST A SECOND REFERRAL IN TRYING TO TRACK SOMEONE DOWN.

The art of referrals is sui generis. This is not the place for a full exposition on the nuances of the form, but they are legion. Simply stated, assume that a referral has an 80 per cent chance of being outdated or simply inaccurate. (Computer simulations supporting this estimate are not available on request.) When in a gambling mood, you may want to ask the person who answers the outdated or inaccurate new number if he or she has an even newer number. Whether you then choose to take a chance on that number is a matter of personal taste.

Corollary A to Law #2:
LAW #2 DOES NOT APPLY TO INTERNS.

Interns are a different matter. Unless it is a matter of life and death, it is rarely worth attempting to track them down. No one ever knows where they are. When I was an intern, I made it a point to surface only about the times plans were being made for barbecues and Christmas parties. In cases of life and death, do not use the phone. Take a chance (and of course you can improve your chances if you know the habits of the person involved) and position yourself at the drug store or in front of the gatehouse the person is likely to use. You could of course leave a message at the intern office, but this is a real longshot. (Do you really think most career panels would have photos of their interns on the walls if the interns showed up there frequently? Those are wanted posters.)

LAW #3: ANYONE ATTEMPTING TO CALL FANX BETWEEN 0900 AND 1400 DESERVES WHAT HE OR SHE GETS.

No commentary required.

LAW #4: THE PERSON RESPONSIBLE FOR MA BELL'S SLOGAN "THE SYSTEM IS THE SOLUTION" SHOULD BE DRAWN AND QUARTERED.

This is the most frustrating part of the process. Street vendors at least have some control over their fate. They can choose to take a loss on a particular sale in the interest of improving their morale if not their business. But what do you do with a busy signal? You are completely at the system's mercy -- and it has none. Wouldn't it be nice if you could gain access to the switching mechanism and make a deal with it? "Okay, I'll take the worst channel you've got, with the lousiest bit rate or whatever, and I won't complain even if it makes FANX sound 4000 miles away. But I've got to complete this call!"

This unfortunately is not a solution, but there may be another one out there, namely:

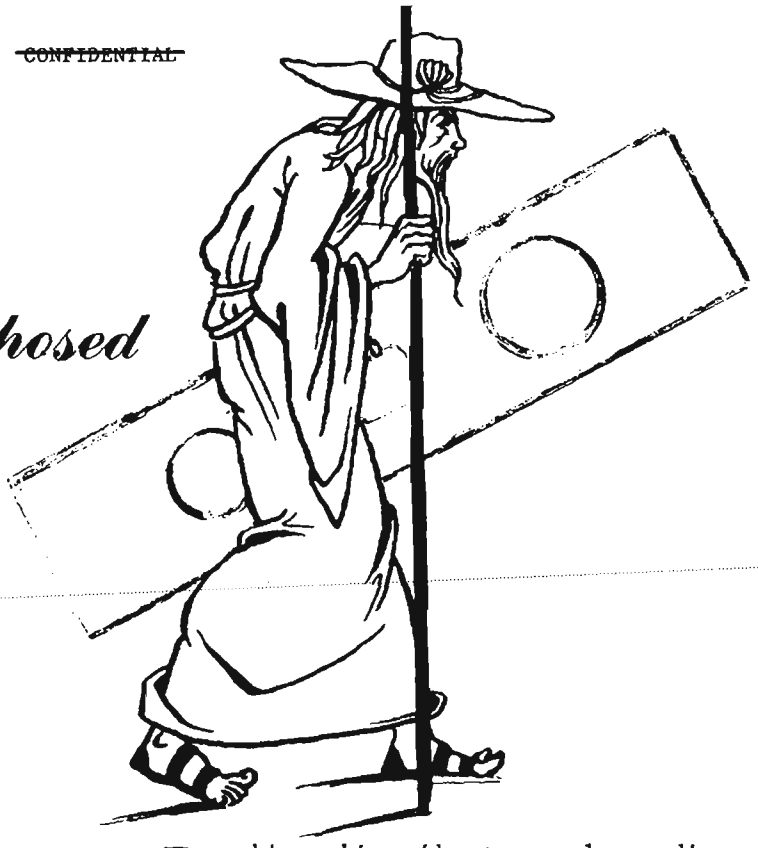
LAW #5: MOST OF WHAT GETS SAID ON THE PHONE DOES NOT HAVE TO BE SAID.

When Oscar Wilde saw his first telephone, the public relations type shilling the thing attempted to impress him by announcing "Mr. Wilde, with this instrument we can talk to Texas." "But what do you have to say?" asked the skeptical visitor.*

---

* The former chief of history and publications insists that this story actually involved Thoreau and the telegraph, not Oscar Wilde and the telephone. Vince Wilson is a learned man. Nevertheless, the story is probably apocryphal in the first place; second, it would be a shame to waste a good line like that on a less than humorous character like Thoreau.

But Life Is *Supposed* To Be Hard (U)



by [redacted] P14

P.L. 86-36

~~(C-CCO)~~ Maybe so, but P14, the TA Division in the Office of Techniques and Standards, is making life a little softer for the working traffic analyst. In preparation are a series of "How to" working aids on the use of T33/P14 UNIX/PINSETTER software for processing, analyzing, and reporting using terminal subsystems. The first in the series, "How to Index", covers the UNIX/PINSETTER commands "index", "permute", and "ptx". Each command is covered in detail and examples are plentiful. The working aids also contain a handy "quick reference". So once a command is mastered, the basic details and variations of the command are available without having to review the text.

~~(C-CCO)~~ Though "How to" working aids are designed specifically for traffic analysts, personnel in related SIGINT fields will also find some of the techniques useful. If you are presently using terminal subsystems for processing, analysis, and reporting, or will be in the not-to-distant future, copies of "How to Index" can be obtained from P14 (room 8A177, phone 3369s). A second working aid, "How to Sort" is being drafted and should be available soon. The following text, with minor adjustments for CRYPTOLOG format, has been taken from "How to Index".

This working aid assumes only a rudimentary knowledge of UNIX/PINSETTER software on the part of the analyst. Consequently, each technique and the (U) command lines necessary to accomplish that technique are covered in detail and examples provided. The examples are basic and unrelated to any specific problem. By studying, and perhaps duplicating the examples shown, the command and options can be adapted to individual target areas.

(U) Points to remember when using this document:

- ★ Examples shown are in the DELTA DATA 7000 (type f) keyboard format. Keyboards for other equipment connected to UNIX/-PINSETTER software may differ, but generally have keys providing similar functions. Operational areas probably have equivalency charts if other equipment is being used. If not, Unit III - The RAND Editor - in the MP119 course (Introduction to UNIX), may be of help.
- ★ UNIX systems sometimes vary from one host to another. These examples were developed on the UNIX host BARDOLPH1, and should work on most other hosts.
- ★ Within the working aid, the symbols [] and <> are used. Data contained in [] (e.g.: [i ignorefile]) denotes use is optional. Capitalized data contained in <> (e.g.: <BREAK>) is required and indicates a specific key on the DELTA DATA 7000

keyboard. In some cases a specific key may also be referred to as the "__" key (e.g.: the ">" key).

★ File names used in the examples and synopses are descriptive only (e.g.: "filename" means the name of a file; "newfile", the name of a new file to be created, etc.) and not a word or a specific file name that has to be used or included when entering a command.

★ For the sake of brevity, some examples may be truncated after the first few lines. Where this is done, it will be indicated by: /////etc/////

★ The word "CAUTION" is occasionally used in the appendix. It is used sparingly and only where necessary to alert the user to a situation that could result in the loss of data or where there are peculiarities in the system that need to be brought to the users attention.

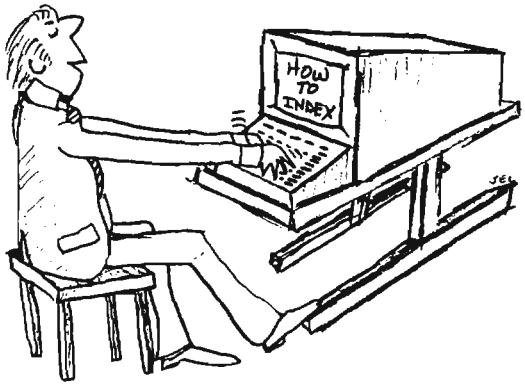
★ Make sure files are "untabbed" before being run through any indexing program:

% untab <filename >newfile <RETURN>

QUICK REFERENCE (U)

COMMAND	OPTIONS	FUNCTION/COMMENT
index	[-]	only one "-" symbol per command line
	[c]	lists words only
	[u]	unique words
	[w]	words, filename, line
	[i ignorefile]	words to be omitted
	[o onlyfile]	words to be indexed
	filename	name of input file
permute	-ainputfile	file name of input file
	[-boutput file]	name of output file
	[-iindicative data]	numbers of columns appended to each line
	[-wpagewidth]	nr of columns (78, 106, 132)
	[-sstart zone & -eend zone]	start & end columns for indexing (zoning limits: page width, - indicative data, - 2, divided by 2, and rounded down)
	[-ddelimiter]	specific character for indexing
ptx	inputfile	name of input file
	[outputfile]	name of output file

Figure 1



INDEXING (U)

DESCRIPTION (U)

(U) Several methods to index data are available to the analyst. The commands INDEX, PERMUTE, and PTX are all excellent tools to index and arrange data to specific individual requirements. By using these commands and related options, the traffic analyst can employ UNIX/PINSETTER software to quickly arrange and exhibit technical data in numerous ways.

APPLICATIONS (U)

~~(C-CCO)~~
of output c
mands used to

☆ "INDEX" works only
special characters are
output. Consequently, this command is best used when digits and special characters are either not a concern or need to be suppressed. "INDEX" works on the entire file and specific segments of a file can not be indexed. It is possible to index only selected words or to omit selected words by using special files with the INDEX command.

☆ "PERMUTE" handles letters, digits and special characters and can be formatted to index only a selected part of a line. For those reasons, it will probably have the most applications for files containing intercept data. There are no provisions to suppress or index selected words, but a delimiter can be used to generate an index.

☆ The "PTX" command generates an index of all special characters, digits and words in a file. Like the "INDEX" command it works only on the entire file. There is no provision to suppress or index only selected words.

EO 1.4.(c)
P.L. 86-36

THE INDEX COMMAND (U)

NAME (U)

index

SYNOPSIS (U)

index [- [c] [u] [w] [i ignorefile]
[o onlyfile]] filename

BASIC COMMAND DESCRIPTION
AND EXAMPLE OF USE (U)

(U) The index command generates an alphabetic index of significant words from a file. When used without options, the index command omits selected words such as articles and conjunctions (e.g. a, an, and, but, the, to); treats capital letters as lower-case; disregards numbers or special characters; and produces an output in this format:

word filename line-nr context

For example, using this sentence as a file and naming that file "filename", the command format and output of data would appear as:

% index filename <RETURN>

The output of data from this command is shown in figure 2.

VARIATIONS IN THE COMMAND LINE
AND EXAMPLES OF USE (U)

(U) As shown in the SYNOPSIS, there are several options (indicated by the data in []) that can be used with the index command. As a general rule, the "c", "u", and the "w" options are used independently, but can be combined with the "i" and "o" options to tailor the output display desired. The functions of these options and examples of their use are:

THE 'c' OPTION (U)

(U) This option lists only the words. A *count* of each word can be made by piping (i.e.: an upper-case backward slash "|" is referred to as a pipe; this causes the system to route your output through another program) this option through the "uniq" command. Again using "filename", the command lines and outputs would be:

% index -c filename <RETURN>

with the output:

```
a
and
and
appear
as
as
command
data
example
file
file
filename
For
format
naming
of
output
sentence
that
the
this
using
would
```

Figure 3

% index -c filename|uniq -c <RETURN>

with the output:

```
1 a
2 and
1 appear
2 as
1 command
1 data
1 example
2 file
1 filename
1 For
1 format
1 naming
1 of
1 output
1 sentence
1 that
1 the
1 this
1 using
1 would
```

Figure 4

appear	filename	3	data would appear as:
command	filename	2	filename", the command format
data	filename	3	output of data would appear
example	filename	1	For example, using
file	filename filename	1 2	sentence as a file and naming that file "filename
filename filename		2	file "filename", the command
format	filename	2	command format
naming	filename	2	naming that file
output	filename	3	and output of data
sentence filename		1	using this sentence as a file
using	filename	1	example, using this sentence

Figure 2

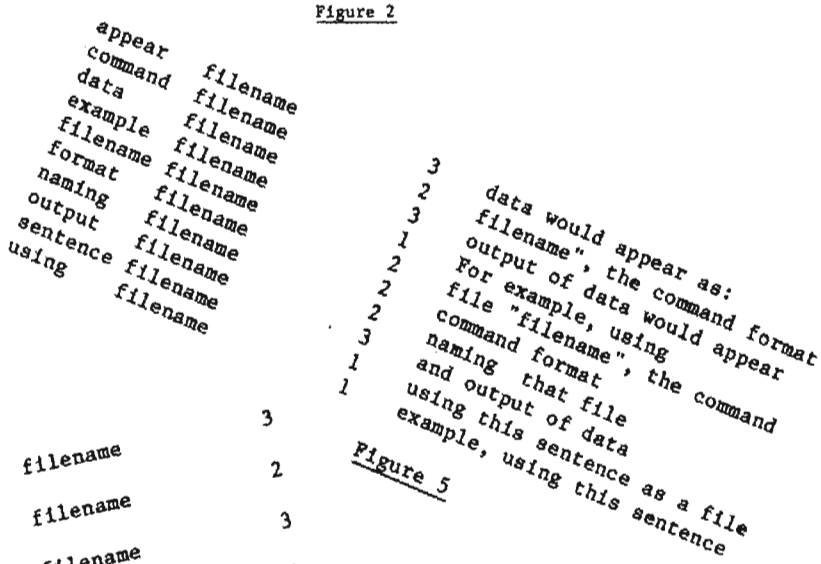


Figure 5

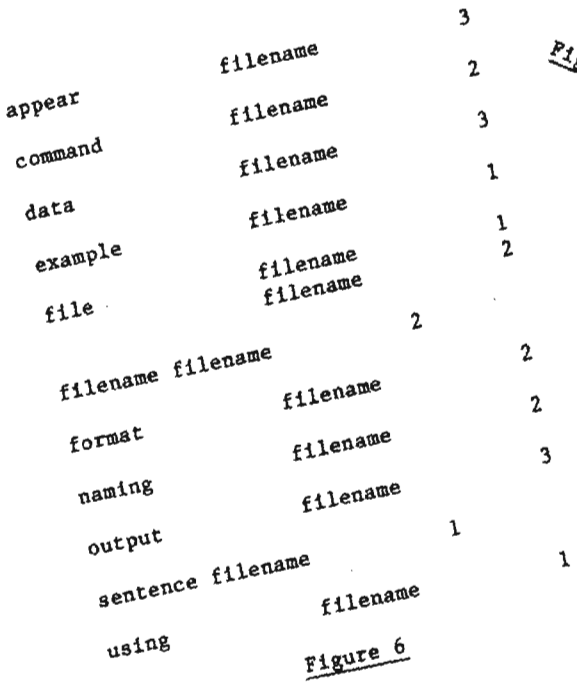
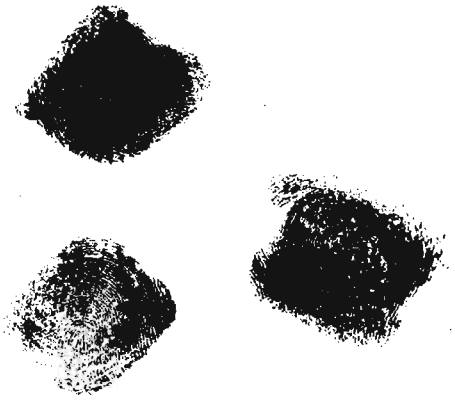


Figure 6



THE 'u' OPTION (U)

(U) This option will list *unique* words. Words occurring more than once are not listed and capitalized words are considered unique from their lower-case counterpart. Using the file "filename" the command line and output (figure 5) would be:

```
% index -u filename<RETURN>
```

Note: The first column of output data is limited to seven letters, if the word being indexed is longer than seven letters, the next column (the name of the file) is offset. Actually, the index output contains a 'tab' character between the first and second columns of data. If the first column contains a word whose length (plus one for the following space) is longer the first tab setting, the rest of that line will be offset one (or more) tab stop(s) (usually eight characters per stop) to the right.

THE 'w' OPTION (U)

(U) This option lists words *without* the context. The command line and output (figure 6) using "filename" are:

```
% index -w filename<RETURN>
```

THE 'i' OPTION (U)

(U) This option must be used in conjunction with another file containing words to be *ignored* by the index command. For example, if you do not want the words "a", "would", "appear", "data", "example", "for", "naming", "of", "output", nor "this" to be indexed, first make a file called "ignorefile" containing those words. Now, using the "i" option and the "ignorefile", the index command will ignore listed words while constructing an index. Below are examples of how to construct an "ignorefile"; the command line used with the "i" option/"ignorefile"; and an example of the output (figure 7). To construct an "ignorefile" use the "cat" command and list, one per line, each word to be ignored:

```
% cat > ignorefile<RETURN>
a<RETURN>
would<RETURN>
appear<RETURN>
data<RETURN>
example<RETURN>
for<RETURN>
naming<RETURN>
of<RETURN>
output<RETURN>
this<RETURN>
<CTRL-d>
```

Note: <CTRL-d> means to depress the CTRL key while hitting the "d" key. This closes the ignorefile and returns the cursor to the screen.

```
% index -i ignorefile filename<RETURN>
```

The "i" option may also be used in conjunction with other options, but unlike most other UNIX or PINSETTER applications, the options are not listed separately (e.g.: -i -c ignorefile filename), but must be combined (i.e.: -ic ignorefile filename). Failure to do so will result in an error listing on the screen. Also remember that whenever the "i" option is used with another option, this combination of options must be followed by an "ignorefile".

```
and             filename        1    file and
                filename        3    and output of data would appear as

as              filename        1    sentence as a file
                filename        3    and output of data would appear as:

command         filename        2    the command format

file            filename        1    as a file and
                filename        2    that file "filename

filename filename          2    file "filename", the

format          filename        2    command format

sentence filename          1    using this sentence as

that            filename        2    naming  that file

the             filename        2    filename", the command

using           filename        1    For example, using this sentence
```

Figure 7

```
and

data            filename
                filename
file            filename        1
                                3    file and
sentence filename filename
                                3    and output of data
                                1    and output of data would appear as:
                                2    sentence as a file and
                1               naming that file "filename", the command format
                                For example, using this sentence as a file
```

Figure 8

THE 'o' OPTION (U)

(U) This option is the opposite of the "i" option. It must also be used with a separate file or "onlyfile" that causes the "index" command to index *only* those words desired. An "onlyfile" is created in the same way as the "ignorefile". For example,

```
% cat > onlyfile<RETURN>
and<RETURN>
data<RETURN>
sentence<RETURN>
file<RETURN>
<CTRL-d>
```

will create an "onlyfile" containing the words "and", "data", "sentence" and "file". Now, using the "o" option, "onlyfile" and "filename", the command line and output (figure 8) would be:

```
% index -o onlyfile filename<RETURN>
```

Like the "i" option, the "o" option may also be used in conjunction with other options to achieve the desired screen output. When used in conjunctions with another option, the two must be combined (e.g.: -oc otherfile filename) and not listed separately.

COMMENTS, OBSERVATIONS AND SUGGESTIONS (U)

(U) You will probably want to save the results of the data indexed. To do so, simply add a "divert" symbol (">" key) and a new file name to the command line. e.g.:

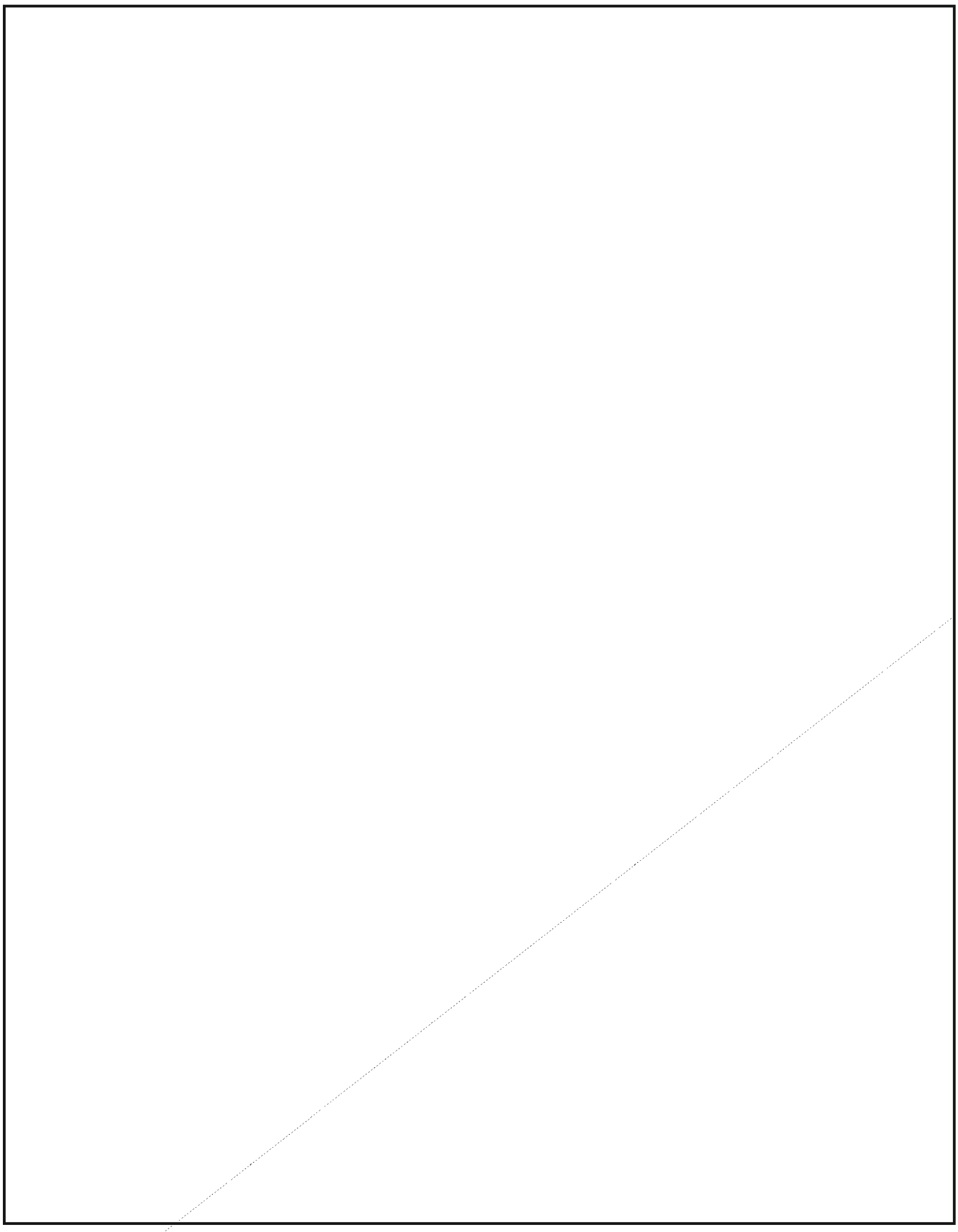
```
% index -c filename > newfilename<RETURN>
```

will place the results of your index of filename into a new file called "newfilename".

(U) Some index commands take a while to run. By adding an "&" (ampersand -- uppercase 6) to the end of the command line, the program will be "run in the background", thereby freeing the terminal for other work. Three CAUTIONS are in order regarding the use of the "&".

- ☐ Do not "logout" while a program is running in the background. Use "detach" to get off the terminal. Otherwise, the system drops your job.
- ☐ Never run more than one job in the background at the same time.
- ☐ Always use a "divert" file when using the "&" option. If you do not, the results return to the screen immediately after completion. This, depending on what you are then doing, could be somewhat undesirable....

~~CONFIDENTIAL~~



~~HANDLE VIA COMINT CHANNELS ONLY~~

~~CONFIDENTIAL~~

THE PERMUTE COMMAND (U)

NAME (U)

permute

SYNOPSIS (U)

```
permute -ainput file [-boutput file]
   [-iindicative data] [-wpagewidth]
   [-sbegin zone & -eend zone] [-ddelimiter]
```

BASIC COMMAND DESCRIPTION
AND EXAMPLE OF USE (U)

(U) Index order is: special characters; numbers; and letters. Upper- and lower-case letters are treated separately, but within the same alphabetical order.

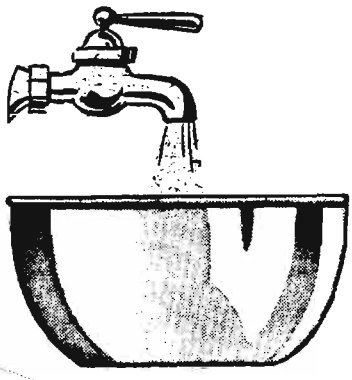
(U) We will use two files, called "filename1" (figure 9) and "filename2" (figure 10). Filename1 contains evenly formated callsign data, while filename2 contains unevenly formated message text. These two files are used as examples to demonstrate the basic "permute" command and options.

(U) Without options, the basic "permute" command functions like the "ptx" command except that all words are indexed. For example, using the "permute" command and the file "filename1", the command line and output (figure 11) look like:

```
% permute -afilename1<RETURN>
```

VARIATIONS IN THE COMMAND LINE
AND EXAMPLES OF USE (U)

(U) As shown in the SYNOPSIS, there are several options (indicated by data in []) that can be used with the permute command. By using these options, the output can be organized to the users specifications and displayed in several different ways. The functions of these options and examples of their use are:



P.L. 86-36

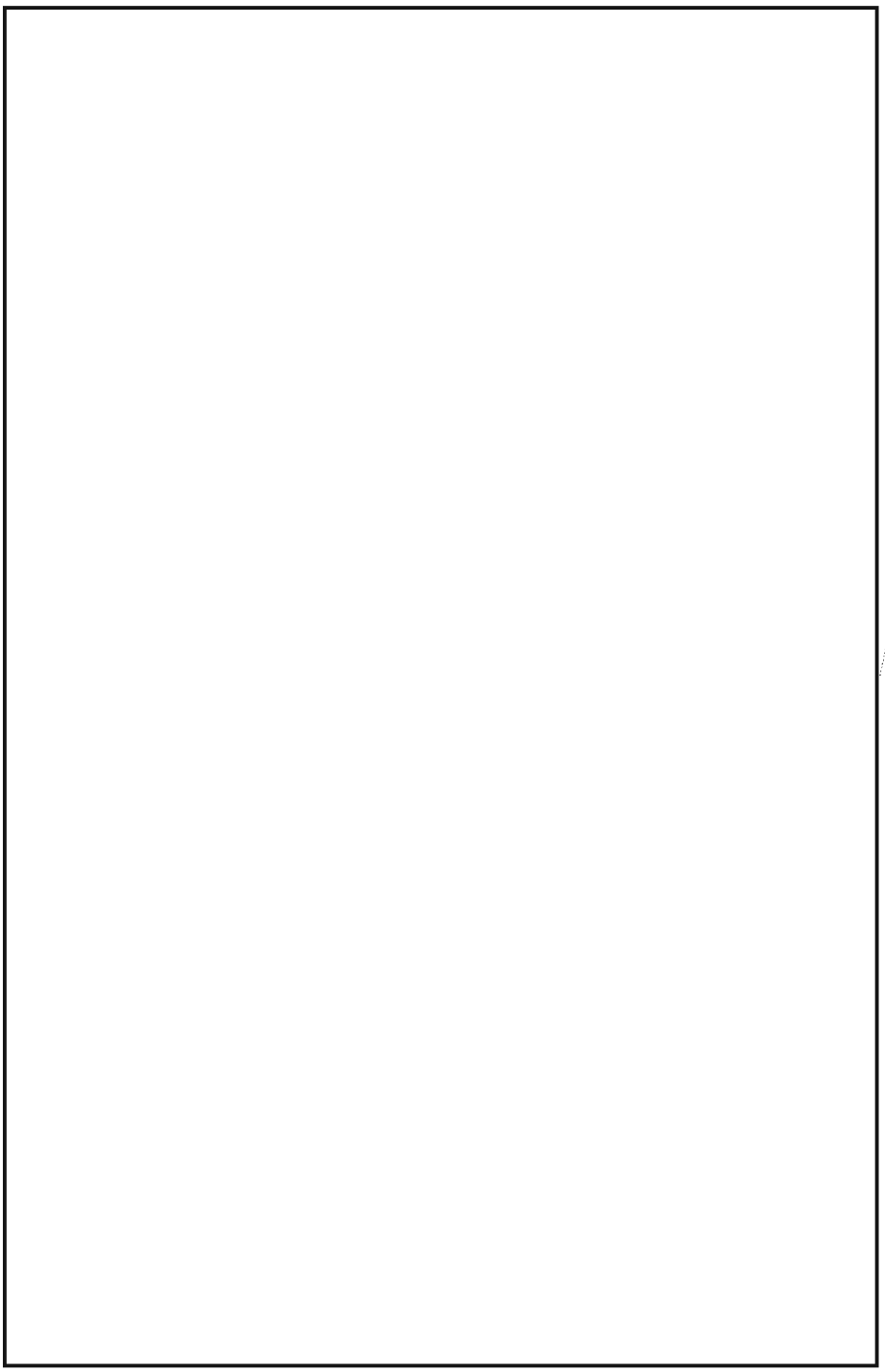
THE '-boutput file' OPTION (U)

(U) Though an election, it is likely that this option will be a part of most command lines when using "permute". The "-boutput file" works exactly like the divert (">") symbol in that it directs the results of the permute index into a file within the working directory. The "-b" part of the option is always followed (no space) by the name of the file (either an existing or new file to be created) in which results are to be placed. CAUTION must be exercised so output is not directed into an existing file containing data, since data already in that file will be overwritten.

(U) Below is an example of the use of the "-boutput file" option. For this example, the output file will be named "text1" and the input file "filename1" will be used.

```
% permute -afilename1 -btext1<RETURN>
```

After the command line is entered into the system, the following will appear on the screen:



EO 1.4.(c)
P.L. 86-36

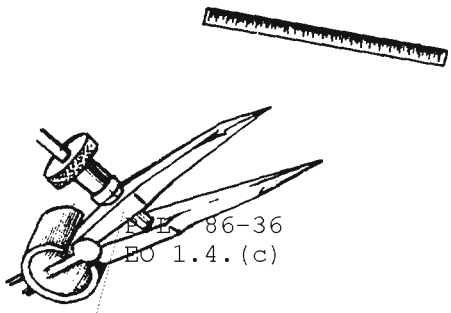
```
SELECTIONS MADE:
NO INDICATOR LENGTH SELECTED
PAGEWIDTH =>78
NO ZONING SELECTED
DELIMITER SELECTED IS SPACE CHARACTER
INPUT FILE =>filename1
OUTPUT FILE =>text1
```

Note: When the job is completed the prompt symbol ("%") returns to the screen. In the above example, a permuted index of material contained in "filename1" would then be in the working directory in a new file called "text1". Since no other options were included in the command line, the output contained in "text1" will be exactly like the example shown for the basic command.

(U) Use of the "-boutput file" option alone or with other options always results in a listing of "SELECTIONS MADE" appearing on the screen after the command line is entered into the system.

THE '-iindicative data' OPTION (U)

Return of the prompt symbol ("%") to the screen tells you the job has been completed and the results placed in the "text2" file. Results would be exactly as shown in the previous example.



THE '-wpagewidth' OPTION (U)

(U) This option sets the page width to either 78, 106 or 132 characters. Widths of 106 or 132 characters must be specified (i.e.: "-w106" or "-132"). If the "-wpagewidth" option is not included in the command line, the page width is automatically set, by default, to 78 characters -- the length of the screen. If the output of the program is longer than the specified pagewidth, the output will be truncated at the specified pagewidth, but the system will try to include the data by "wrapping it around" or fitting it to the other end of the output line. This is indicated by the indicator "..." and should be apparent in the text. In some cases with longer records, data will be lost unless a longer page width is specified.

(U) One of the reasons for and use of the "-wpagewidth" option becomes apparent when "filename2" is "permuted" first with a page width of 78 and then again with a page width of 106 characters. For example, a pagewidth of 78 characters (which is obtained by default when not specified) would be obtained by the command line (output in figure 13):

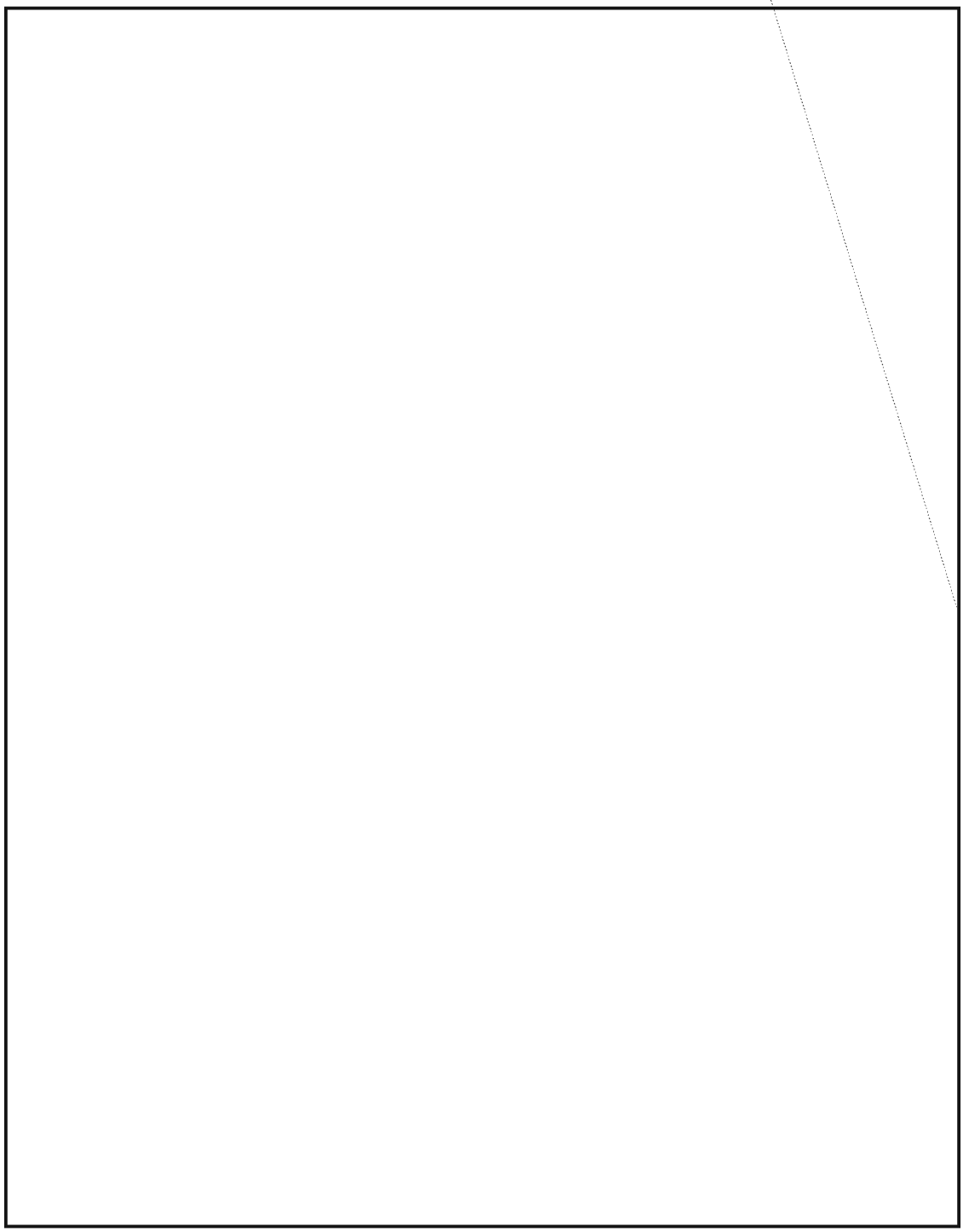
```
% permute -afilename2<RETURN>
```

Even though no data was lost (the input record was only 78 characters), some lines of text are "wrapped around" as indicated by the "..." at the end of text. Now, using an option of 106 characters, the command line (output in figure 14) would appear:

```
% permute -afilename2 -w106<RETURN>
```

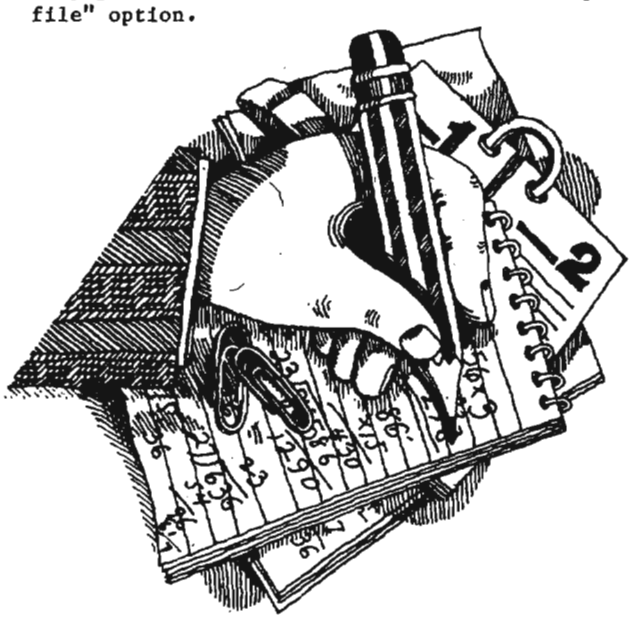
Note that some text is no longer "wrapped around" and that the symbol ">" now appears at

EO 1.4.(c)
P.L. 86-36



the end of some lines. This indicates additional text exists on the line, but is not shown because of screen size. If in the RAND editor, hit the <PORT-right arrow> key and the screen will move over to the text. To return the screen to its original location, use the <PORT-left arrow>.

(U) Another reason for the "-wpagewidth" option will become apparent with use of "-sbegin zone" and "-eend zone" options discussed below. Like all other options, the "-wpagewidth" can be used with the "-boutput file" option.



THE '-sstart zone & -eend zone' OPTIONS (U)

~~(C-CCO)~~ The permute index can be restricted to a specific portion of the input record. This is called "zoning" and requires that the start and end columns or "zone" be specified in the command line (e.g.: "-s35 -e47"). The "-s.." and "-e.." are interdependent and must be used together. For example, if you wanted to restrict a permute index to control callsigns (columns 19 through 22) in "filename1", the command line (output in figure 15) would look like this:

% permute -afilename1 -s19 -e22<RETURN>

(U) The size of the "zone" that can be indexed is limited. If the standard 78 columns are used, then the "zone" is restricted to a maximum of 38 columns. If the 106 or 132 column page width is specified through use of the "-wpage width" option, the "zone" maximums become 52 and 65 columns, respectively. If a "zone" greater than the maximum allowable is specified in the command line, the program will abort and tell you, via the screen, why the program was not run. For example, the following command, which exceeds the zone limitation (38 columns) for a page width of 78 columns:

% permute -afilename1 -s10 -e60<RETURN>

causes the error line below to return to the screen.

Can't fit zoned input & indicator on page
Execution terminated

(U) If an "-iindicative data" option is used, the size of the "zone" is further reduced. Simple math can be used to determine the maximum allowable "zone" when an "-iindicative data" option is included. The formula is:

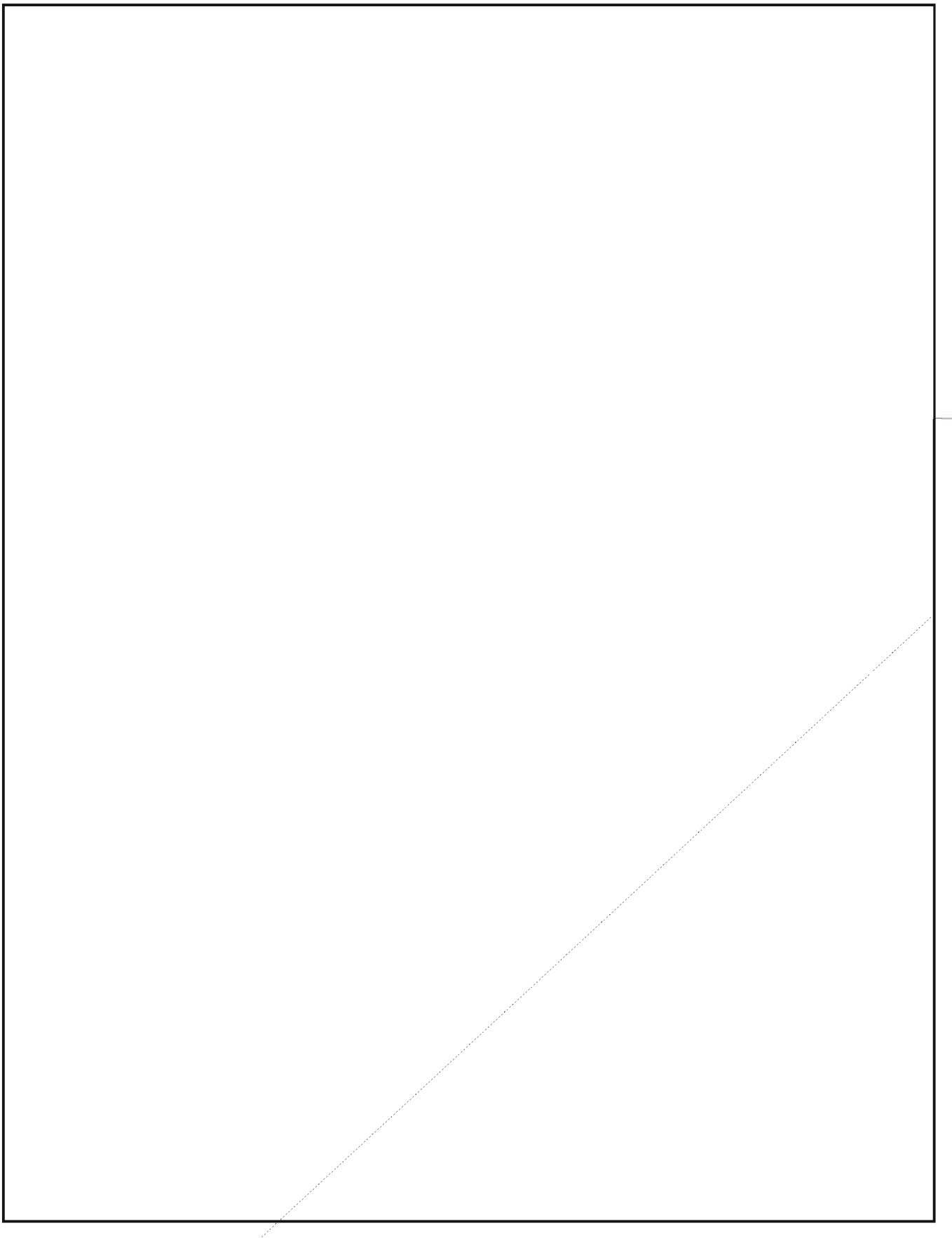
(page width)-(indicative data)-2, divided by 2 (fractions of a column are rounded down to the next whole number).

For example, if the page width is 106 columns and the indicator is 17 columns, then the maximum allowable "zone" is 43 columns (106-17-2)/2 = 43 1/2, which is rounded down to 43. If the combined "indicator" and "zone" exceed the limit for the page width, the program terminates and an error message returns to the screen.

THE '-ddelimiter' OPTION (U)

(U) This option is especially useful when working with data that does not have a set length or when you want to have only a certain item (such as all text beginning with a "3") indexed. The term "delimiter" used here refers to a specific character (e.g. "/", "(", "a", "6", "Z", etc). By default, if the "-ddelimiter" option is not included in the command line, a space is considered to be the "delimiter".

(U) When a specific character is used for a "delimiter", special rules apply. It should be used either with zoning or with increased page width. Failure to do so may result in the program being terminated, an error message appearing on the screen, and a "core" being placed in your working directory. If this happens, first remove the core (% rm core<RETURN>) and then either specify a zone (e.g.: -s30 -e40) or increase the page width (e.g. -w132), or both, in the command line.



(U) Following is an example of the use of the "-ddelimiter" option being used with "filename2". In the example, the message text (columns 32 to 63) is zoned, and the "/" symbol is used as the "delimiter". The example is brought back to the screen, but would normally be placed in another file by using the "-boutput file" option (see figure 16).

% permute -afilename2 -s28 -e63 -d/<RETURN>

Note that in the above example:

- The program automatically placed "/" characters before the first column of data in the zone. This allowed the first part of the message text, which was not preceded by a "/", to be included in the index; and,
- [redacted] If an "indicative data" option had been included in the command line and also (because the "indicative data" and "zone" together exceed the limit for the page width of 78 columns) the page width increased, this data would have been retained.

(U) Now, including the "indicator" and "pagewidth" options, the entire command line and output (figure 17) would look like:

% permute -afilename2 -s28 -e63 -d/ -i27 -w106<RETURN>

All the indicative data is now shown and the page has been expanded to 106 columns. The file is best viewed in the RAND editor where the <PORT> keys can be used to see the entire text. (If your terminal does not have <PORT right arrow> and <PORT left arrow> keys, you can get the same action by using <Ctrl s> for <PORT right arrow>, and <Ctrl a> for <PORT left arrow>).

COMMENTS, OBSERVATIONS, AND SUGGESTIONS (U)

(U) Like the "ptx" and "index" command "permute" can be "run in the background" by adding an "&" (ampersand) to the command line. When running the "program in the back ground", an output file should also be specified to prevent the program returning to the screen when ever the job completes. With the "permute" command, this is done by including the "-boutput file" option; not by using the "divert" (">") symbol as is done with the "ptx" and "index" commands.

(U) Users should be aware of an idiosyncrasy occurring when the "&" and the "-boutput file" option are used together. Basically, when the command line is entered, the system provides a process identification number (frequently referred to as the process ID or PID) and returns the prompt ("%") to the screen. Immediately thereafter, the "SELECTIONS MADE" listing also returns to the screen. Since the prompt symbol is already on the screen, it does not reappear after the listing completes. Consequently, with the "SELECTIONS MADE" listing between the prompt and the cursor, it is not apparent that the system is ready to accept another command. However, a new command can be entered immediately or, if you prefer, hitting the <@> and <RETURNS> keys brings another prompt symbol to the screen.

(U) CAUTION. Care must be exercised when the "permute" command is used on a large file. In cases where a sizable zone of data in a large file is indexed, the resulting output can exceed available temporary working space and cause your user group system to "crash". This problem is compounded by running the command in the background using the "&" symbol. If necessary, large files can be split into several files using the "split" command; permuted separately; appended together; and than piped through the "csort" command to achieve the desired output. For example, the command line:

% split -250 filename<RETURN>

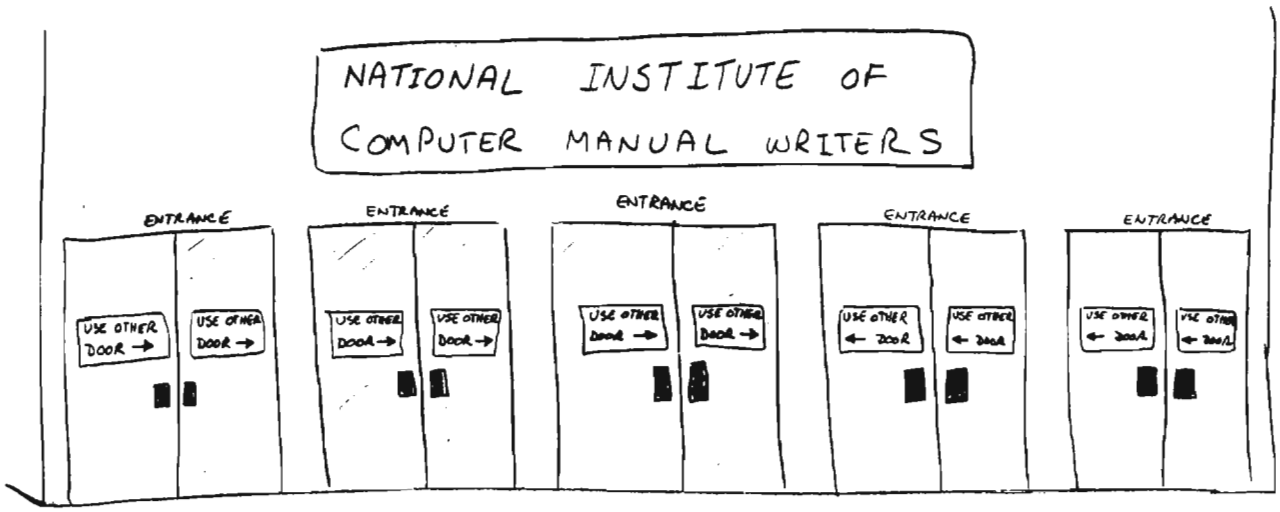
would split the data in file "filename" into separate files each with 250 lines of data. The resulting files ("xaa", "xab", "xac", etc.) can than be permuted separately (but not in the background at the same time). In turn, these files can be combined and the output "piped" (the "|" key") through the "csort" command and its "+col" and "-o" options. The "+col" option specifies the column where the merge is to occur (this will be the same as the start of the zone used in the "permute" command); and, the "-o" option precedes the name of the file in which the results are to be placed. For example the command line:

cat xa xb xc >> f1 |csort +25 -o f2 <RETURN>

would append the files "xa", "xb" and "xc", into "f1" and pipe "f1" through the "csort" command. In turn, the "csort" command sorts "f1" on column 25 and places the results into a file called "f2". Had the start of the zone previously used to permute the three files (in this case +25) been omitted, the merge would have been left-justified and not in the desired format.

```
   as the special characters  !, ", #, and...as well
as the special characters !,  ", #, and...as well
                 command....  $ and was made to demonstrate the ptx
the special characters !, ",  #, and...as well as
        contains the numbers  1, 2, and 3..."filename",
     contains the numbers 1,  2, and 3..."filename",
       the numbers 1, 2, and  3..."filename", contains
 special characters !, ", #,  and...as well as the
  sentence, placed in a file  called...This
     as well as the special  characters !, ", #, and
 made to demonstrate the ptx  command....$ and was
                   The ptx    command line using the file "filename"
               "filename",    contains the numbers 1, 2, and 3
          $ and was made to   demonstrate the ptx command.
  This sentence, placed in a  file called
  ptx command line using the  file "filename"...The
                 and 3...     "filename", contains the numbers 1, 2,
 command line using the file  "filename"...The ptx
       This sentence, placed  in a file called
and the resulting output look like this:
             The ptx command  line using the file "filename"
    and the resulting output  look like this:
                 $ and was    made to demonstrate the ptx command.
   "filename", contains the   numbers 1, 2, and 3
        and the resulting     output look like this:
           This sentence,     placed in a file called
 was made to demonstrate the  ptx command....$ and
                     The      ptx command line using the file "filename"
                  and the     resulting output look like this:
                     This     sentence, placed in a file called
              as well as the  special characters !, ", #, and
               "filename"...  The ptx command line using the file
                              This sentence, placed in a file called
  resulting output look like  this:...and the
       The ptx command line   using the file "filename"
                     $ and    was made to demonstrate the ptx command.
                   and...as   well as the special characters !, ", #,
```

Figure 18



THE PTX COMMAND (U)

NAME (U)

ptx (permuted index)

SYNOPSIS (U)

ptx inputfile [outputfile]

BASIC COMMAND DESCRIPTION AND EXAMPLE OF USE (U)

(U) The ptx command generates a permuted (permute: arrange objects in a series in all the possible ways in which they can be arranged -- Webster's 3rd New International Dictionary) index of all words, numbers and special characters in a file.

(U) The index sort order is: special characters first, digits next, and letters last. Capital letters are treated as lower-case and punctuation ignored. The key word appears in the middle of the page with preceding and following text also shown. The words "a", "an", "and", "as", "is", "for", "of", "on", "or", "the", "to", "and" and "up" are suppressed for key word use, but included in preceding or following text.

This sentence, placed in a file called "filename", contains the numbers 1, 2, and 3 as well as the special characters !, ", #, and $ and was made to demonstrate the ptx command. The ptx command line using the file "filename" and the resulting output look like this:
(see figure 18)

% ptx filename<RETURN>

VARIATIONS IN THE COMMAND LINE AND EXAMPLES OF USE (U)

THE 'outputfile' OPTION (U)

(U) The "output file" option will be used frequently. It places the results of the ptx index into a new file within the working directory. This is done simply by adding an additional file name (e.g.: "newfile") to the basic command line. For example, the command line:

% ptx filename newfile<RETURN>

will ptx the data in the file "filename" and place the results into another file called "newfile". Because of the way the program was originally written, this is an automatic "divert" and it is not necessary to place a "divert" symbol (the ">" key) between "filename" and "newfile". Use of the "divert" symbol will; however, produce the same results.

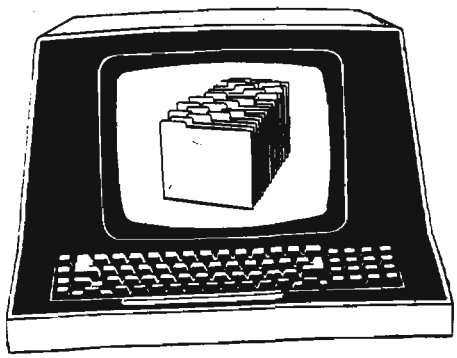
COMMENTS, OBSERVATIONS AND SUGGESTIONS (U)

(U) When using the ptx command it is frequently advantageous to use the "divert" option and "run the program in the background" in order to free the terminal for other work. To do so, simply add an "&" (ampersand -- upper-case 6) to the end of the command. For example:

% ptx filename newfilename &

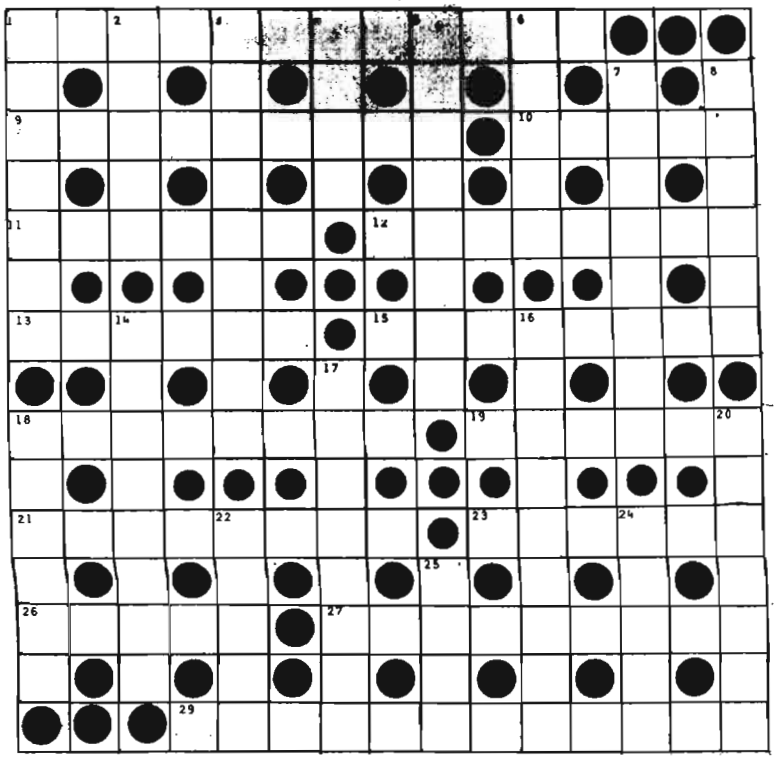
CAUTION: If you leave the system while a job is running in the background, use "detach" vice "logout". Otherwise, the job is dropped. Also, make sure there is a "divert" file specified in the command line. Failure to do so, creates instant garbage when the job completes, returns to the screen, and combines with whatever you are doing at the time.

(U) Before running a ptx, it is frequently best to do a little selective editing to remove useless lines.



CRYPTIC CROSSWORD

By [redacted] P.L. 86-36



ACROSS

1. Do similar growers produce bugles? (7,5)
9. Closemouthed as a result of severe tic? Awful! (9)
10. The glove let out? It's too thin! (5)
11. Believe me! Tick fever symptoms include cause to vomit! (6)
12. Lunatic pets goat and holds up exit. (8)
13. Never consult a nautical chart unless holding a ruler! (6)
15. The congressman gave public utterance and made amends. (8)
18. Settles one charged particle in among equals. (8)
19. Aha! Include a gun for Christie. (6)
21. Come back again to harvest fruit? (8)
23. Innate order where one finds fish. (2,1,3)
26. The Spanish and the German berry bush. (5)
27. Is the lout suing over the sticky stuff? (9)
28. A nice insider goes berserk over bombs! (12)

DOWN

1. For samplers, take some blossoms after tea. (7)
2. Relative expression of surrender? (5)
3. Affirm recipe that puts apple juice back into chopped liver? (9)
4. Double or nothing out of race objective? (4)
5. A wet rice concoction for a summer beverage? (3,5)
6. Wild glee about a fine golf score! (5)
7. Sovereign trouble initially after car crashes into car! (8)
8. London art gallery moved into South Dakota, it's said. (6)
14. Horrible! Drop sale of cats! (8)
16. Man enters a rain, oddly, in South America (9)
17. Attempt, by the sound of it, to fish for a polygon. (8)
18. Insane report carries the load! (6)
20. Swears to student's whereabouts at end of term, perhaps. (7)
22. An anonymous tipper once exposed a former dictator. (5)
24. She comes from main dispersion around the first of October. (5)
25. Being impolite, we hear, is to be regretted. (4)

Rules for the Camel Corps (U)

by [redacted] T095

It is frightening to contemplate the amount of time NSA employees spend in meetings. There are staff meetings at all levels, working groups, meetings to solve a particular problem, club meetings, and meetings to find reasons for more meetings. "He's at a meeting" is all too frequently heard on the other end of a phone line.

As a veteran staffer, I feel qualified to say that not all of this is time well spent. I have fidgeted and chafed through many meetings that were unnecessary, poorly run, overlong, and ended in chaos. The people who run these I have affectionately labeled the "Camel Corps" because their product so often resembles the design of the proverbial horse. Conducting a proper meeting should not be terribly difficult as long as the chairman follows a few simple rules.

1. Make certain a meeting is really necessary. A few strategic phone calls or memoranda may provide a more efficient solution to the problem.

2. Schedule the meeting for a fixed period of time. Except under very special circumstances, no meeting should last longer than one or one and a half hours. By that time, you've lost the narcoleptics and doodlers and are in danger of losing everyone else. It's better to schedule two sessions than to have one become unproductive.

3. Invite <u>all</u> the right people and <u>only</u> the right people. It's all in vain if you're missing what the Camel Corps would call "a key player." Also, people who have no reason to be at a particular meeting tend to become hostile or to lead the discussion into an area where they can have input. Work with the smallest possible group.

4. Prepare an agenda or discussion outline and distribute it in advance. Everyone should be aware of the contribution he is expected to make. If possible, include background information and ensure that pertinent documents are available to all.

5. Keep the discussion on the subject and under control. Digressions and "war stories" should be nipped in the bud. This is an important way to help accomplish Rule 2. Above all, don't let someone else assume command of your group. I find that most people respond very favorably when a meeting is effectively kept on the subject at hand.

6. Give everyone a chance to express his opinion. The person sulking in the corner may have a more important comment than the most outspoken attendee.

7. If you finish early, dismiss the meeting. I remember one leading Camel Corps member who accomplished his stated purpose in the first five minutes of a scheduled one-hour session. This disconcerted him, so he kept up a pointless discussion just to fill time. I walked out.

8. Briefly summarize the session before ending it. This reduces the chance of later misunderstandings. Each participant should be aware of any follow-up actions for which he is responsible.

9. Prepare a written record for participants and for anyone else who should be "brought up to speed" (Right, more CC slang!) on the meeting. Keep the record yourself until all danger of repercussion is past.

Sound too simple and obvious? Think about them whenever you take part in a meeting. I don't think it will take you long to learn to spot the Camel Corps at work.

TOWARDS BETTER SYSTEM DEVELOPMENT (U)

by [redacted] R53

This paper was originally written for the first annual conference of the Communications Analysis Association on communications analysis: "The Analyst in the 80's." It appears here with minor revisions.

The Analyst in the 80's will continue to be seriously affected by personnel constraints, changes in the workforce, and the evolution of the communications and technological environments. Among the various approaches that will be used to deal with these problems will be an increasing use of tools to support the analysts in their various activities. Central to these tools will be interactive computer systems.

Personnel constraints that limit the size of the workforce are a source of pressure for increasing productivity. Changes in the workforce caused by the retirement of experts and the difficulty of recruiting and developing their successors are a source of pressure for finding ways to capture existing expertise for transfer to future generations. Evolution of target environments to more sophisticated techniques and greater capacity is a source of pressure on diagnosis, the development of attacks, and their transfer into computing systems. Responses to such evolution range from use of a known attack on a larger scale, refinement of a known attack, development of a new attack, and various combinations. Continued development of more useable, available, and powerful computing resources for Analysts is the fundamental means of automating the more mechanical parts so that Analyst resources can be focused on the more intellectual.

For example, some known "old hand" attacks may be just what is needed for some "new" problem. However, effective exploitation and utilization of personnel may require the development of new computer systems. Although the "old hand" attack might already seem to be sophisticated when it is carried out by a "human processor," additional sophisticated techniques may be required to move it to a "computer processor." It may not be possible to move all of the human processing to a computer because some aspects of the human processing may not be well enough understood to be mechanizable. This requires a close coupling between intelligent human and mechanical computer processors to produce a complete system solution. As the target problem becomes more difficult in techniques or scale the system development activity becomes correspondingly more difficult.

Approaches to deal with these pressures share the idea of developing facilities to support the Analysts. The facilities can be viewed as tools that can be applied to various parts of a problem. A tool provides a way to improve the interface between the human problem solver and the problem. Those parts of the problem which are well understood can be automated so that valuable Analyst time can be devoted to the more interesting parts. This is a continually-evolving process which raises the level at which the Analyst operates. Suppressing the details that are no longer of interest exposes new and more interesting problems that were previously hidden.

Responses to the pressures are in the form of computer systems supporting increasing amounts of interaction. In time these systems will evolve to become environments which will support almost all of an Analyst's activities. This kind of response transfers pressures on the Analysts to pressures on the system Developers. The general approaches used to respond to Analyst pressures can also be applied to system Developer pressures.

In the remainder of this paper the term "User" will refer to any person using a system to assist him in carrying out some activity. Since we will be addressing system development for Users, a brief sketch of system development from a User's view is useful to establish some context for what follows. From the User's view, system development starts with some notions of a need for a system to formulate system requirements. After the requirements are formulated, the system is developed, and eventually delivered so that it is available to the Users. To the User, the system development activity looks like a delay between the formulation of requirements and the system's availability. Although the Users may believe that the requirements embody their notions of what they need, and the Developers may believe that the system developed satisfies these requirements, the delivered system often contains some surprises.

WHY?

Before trying to answer this, some discussion of the kinds of systems we are evolving to will be useful.

Tools and their Environments

Just as tools have been and will continue to be developed to help the User solve problems, the development of the tools themselves is a problem requiring tools. The development of requirements for a system to solve some problem for a User is clearly a serious problem in its own right. This is evidenced by the difficulty in deciding what problem is in need of a tool, what kind of tool needs to be developed, how the tool is developed, how it will be used, who will use it, where it will be used, and the like. In fact, the very existence of a tool for a problem and experience using it often provide useful and critical insight into the problem and expose the need for new tools. Some general properties of tools and their importance include the following:

- Tools are not used in isolation; they may be used in combination with other tools.
  - ☐ It is important that they can be combined into larger ones. Developing tools which can be combined to form new tools is fundamental to the ability to build upon previous work rather than repeating it.
- Tools interact with the User; they may not be completely automatic.
  - ☐ It is important that excellent human factors be a major design objective in their development.
- Tools interact with a variety of data, information, and knowledge sources.
  - ☐ It is important that these sources are an easily accessible part of the environment.
- Tools may be applicable to a class of problems which differ in only some specific parameters.
  - ☐ It is important that given the identification of the parameters, it is possible to develop a single parameterized tool to insure the consistency of the different members of the class.
- Tools need to evolve as the problem and approaches to it evolve.
  - ☐ It is important that the tools can be evolved rapidly to maintain their applicability.

> *systems have no intrinsic value*
> *except to the extent*
> *that they satisfy their users*

user expectations are rising
as a result of
previous development successes and
the decreasing cost of system components

The development of tools that satisfy these properties is itself a problem that can benefit from tools of its own. Central to tools for the Users and the tools to produce them will be the use of interactive computing systems. Time-sharing systems have been used as the basis of most interactive systems, and are reaching limitations in the ability to support adequate human factors and desired functionality. The decrease in hardware costs has reached the point where personal computing with network facilities is an attractive alternative. The importance of a network cannot be overemphasized. Personal computers alone are not a step beyond time sharing if they do not provide other kinds of resource sharing in a transparent fashion. Traditional programming languages and techniques could be used but more advanced ones are emerging which also provide attractive alternatives.

The Requirements Problem

We have seen the trend from pure requirement-generation by User organizations passed on to development organizations evolve to joint User/development requirement-generation. Although this is a step in the right direction, experience has shown that joint requirement-generation does not go far enough. Systems built from joint requirements may not meet the expectations of the participating parties. When the systems are completed and the problems discovered, it is often too late to take necessary corrective actions.

The way in which systems are developed is heavily influenced by the perceived cost of the various aspects of the development activity. The structure imposed on the activity tends to be proportional to the perceived difficulty of development. For some problems that are well understood, requirements can be written by the User community and given to the Developer community. If the resulting product satisfies the requirements, then the Users should be satisfied with it. This is a fictional approach for most problems of interest.

This model takes a simplistic view of the need for communication between Users and Developers. It does not provide for any feedback from the Developers on their understanding of the requirements until the product is delivered. In an attempt to narrow the gap between the Users' and Developers' understanding of requirements, joint User/Developer requirement generation emerged. The rationale was that if both parties are involved in writing the requirement, then the chance of getting the desired product should improve.

Although joint requirement-generation does yield some improvement in the communications area, it still takes a simplistic view of the role of requirements. The generation of requirements assumes that enough is known about a problem to say what is needed. Even if enough is known about a problem, how can requirements be written so that any two people involved with them agree on what they mean? And if not enough is known about a problem, then requirements are even harder to write.

In reality, the problems are sufficiently difficult that not enough is known to write complete requirements, and it is not possible to write them in a way that guarantees uniform interpretation. There have been a number of significant efforts to address the requirements problem, and still it persists. After some period of little progress, one begins to look in other directions. Could it be that there is something wrong with the idea of developing requirements or the way in which they are developed? This will be explored next and alternatives considered.

The Prototyping Approach

Requirements are hard to write, and even when well-written and comprehended by all involved can lead to unexpected results. When

the system is completed and the problems are discovered, it is often too late to take the necessary corrective actions. This is because it may not be possible to change the system without substantial delays. The form of a developed system may be very sensitive to some of its design decisions. Changes to such critical design decisions can have pervasive effects on the developed system so that major parts of the system may have to be modified or developed again. The translation of a system design into a developed system has traditionally been a manual one. In addition the target of the translation, even if it is one of the common higher level languages, often requires detailed design decisions to be made prematurely just to get the system working. These languages do not provide a convenient means of adjusting the level of the target to one that is high enough so that lower level design decisions can be deferred.

As a consequence of these factors, the cost of the software development activity becomes high enough that emphasis is placed on trying to do as much as possible before any software development begins. Although it is always considered a good approach to put effort into preparation, the preparation and performance activities need to be carefully balanced. It is becoming increasingly clear that the usual way in which requirements for a system are developed is not adequate because of the increasing frequency of surprising undesirable results of the translation of requirements into a system. This is an example of too much emphasis placed on a form of preparation which does not adequately identify the important issues early enough. Among these issues is how the system is really expected to behave!

Experience with system development has repeatedly shown that more insight is obtained by the use of a system than by simply trying to read about what is expected of it. Given a system and its requirements, the behavior is more readily evaluated from its use than from thinking about how it might be used. This is not to say that all aspects of system evaluation can best be comprehended through examples. In fact, there are many important properties of a system that cannot be effectively determined by its use. For example, "bugs" can be found by using a system; but it is not (in general) possible to know when all the "bugs" have been found. The point is that a prototype of a system can be a very useful tool during the usual requirement and development activity. To be most effective, prototype development needs to be rapid so that it can be quickly made available to the User for which the requirement was formulated. A product of the requirement activity can be a prototype which is used to start the development activity.

In the prototyping approach, development becomes a prototype refinement activity. Although traditional programming languages and techniques could be used, they have some significant limitations. These limitations are the same as those which led to the traditional system development approach. An understanding of these limitations and a more advanced approach to programming languages and techniques that deals with them can break this cycle. The limitations and their consequences include:

- The low level to which a design must be carried out to produce a working system because the language level is fixed and far below that of the problem domain.
  - As a consequence, more detailed and otherwise relatively irrelevant decisions may have to be made just to get a prototype developed.
- There is no "explicit" form of the design and no way to transform it into a working system except manually.
  - As a consequence, changes in design decisions have to be carried out manually, making changes very expensive and limiting opportunities to learn from experience. A valuable product of the development activity is lost for future redevelopment. The only "real" product is the crystalization of the design. Although this is generally too fragile to be changed, it is the only thing available, short of developing the system again manually.

development capacity
is a limiting factor
in satisfying user requirements

> *if we had perfect insight,*
> *then systems could be developed*
> *from their requirements;*
> *but we don't*

A more advanced programming language and development technique should address these two classes of issues directly. Programming languages are needed which can be easily extended to problem domains characterized by the appropriate abstractions including notation, objects, and operations of the domain. System development environments are needed which can support a multi-level design which captures the design decisions. The combination of these two facilities can support the rapid development of prototypes, the development of designs, their transformations to production systems, and their redevelopment.

The Tension between Users and Developers

When new computer systems are being developed for Users, care must be taken to insure that the resulting system is reasonably close to what is best for the User rather then what is best for the system Developer. Each usage of "best" is relative to a different point of view based on the subject to which it is applied. For example, if the notion of "best" is that of "most cost-effective in solving the problem", the independent interpretation will still lead to conflicting results. This is because the User view considers problem-solving effort of human resources as the most valuable, while the Developer view considers the system itself as a valuable resource. The proper point of view lies somewhere in between these two extremes. Given the pressures on the User and the advancing state of computing technology, it is clear that the point can be put very close to the User in the future. As a result, the situation dramatically changes because system resources have and will continue to dramatically reduce in cost.

Unfortunately this does not remove the tension between the User and the Developer; it just changes its character. The decrease in cost of system components raises the expectations of the User. The problem then becomes one of combining components to meet these new expectations. This results in the problem solving time of Users and Developers in their respective domains becoming the critical resource. The User wants new tools to help solve problems, and the Developer needs to produce those tools. But the development of tools is itself a problem domain which can benefit from tools. The right tools have the potential of reducing development costs so that the Users get their tools sooner. Tools to produce tools could become tools of the User.

Some Familiar Consequences of this Tension

Some consequences of this tension being resolved in a way that is not favorable to the User demonstrate the need to be more sensitive to the User view. This goes beyond the system Developers because, in some cases, new technology is needed to fill important gaps. Initially the benefits of such systems outweigh their limitations. After some experience with them, the User expectations are raised and new issues are exposed. There is a tendency to try to use a given system for broader applications. Unfortunately the existing systems do not scale up well. There are some systems which are much closer to the ideal in some areas than others, but these have not all converged into a complete system yet.

Applications to New Technologies

A popular topic in future computing technology is personal computing. Although it seems clear that this technology offers many advantages over even existing advanced time-shared systems, there is much to be learned about it. At this time, it may be hard even to write requirements for future systems based on new technology because of our lack of experience with it. A useful way to gain experience in a new technology is to set up a laboratory for research in the area. Some early explorations have been done in various places, including R53, the Office of Computer Science Research, which already has a strong foundation in the underlying technologies. An example of such an approach can be found in the paper "A Laboratory for Developing Personal Machines" by [redacted]

Advanced System Development Environments

No matter how advanced the hardware environments may be, they cannot be used effectively without advanced system development environments. An important component of an advanced environment is an advanced programming language. But an advanced language alone is not a sufficient goal. An advanced environment is needed to support the use of the advanced language throughout the lifecycle of the system. Some limitations of most existing environments will be discussed and some indication of what more advanced ones are likely to be like will be given.

The limitation of the traditional programming languages, including the higher level ones, is the size of the gap between the conceptual level of the problem domain and the language. Solutions to problems are expressed as programs. If the programs can be written in a notation natural to the problem domain then the gap is very small. As the problem domain moves further from that supported by a language, more effort is required to reformulate the natural problem-domain expressions to the lower level supported by the language.

If the problem domain is one for which the language is designed, then solutions will be easy to express in the language. However, if the domain involves concepts that are not in the language, then problem solutions will also have to solve a second problem caused by the language itself. The additional language problem may be sufficiently difficult to divert attention from the problem which was being solved in the first place.

Of course, special dialects of some languages have been developed to support more natural notations for this problem domain with these data types and operations. But, suppose that some problem is best expressed in terms of another data type. An unnatural notation will again be needed. This problem exists within the very limited domain of numerical expressions. The non-numerical world has a very rich collection of data types. In many cases they are created for each problem. From a human factors point of view the natural notation is the best. This is an example of the tension between Users and Developers in the domain of programming languages.

Programming languages which are advanced in this way and which can be used in an interactive environment are not just tools for the Developer to produce systems for the User; they can themselves become tools for the User! The importance of the environment must not be underestimated. Even with an advanced language, additional facilities are needed to support its use. These facilities need to have excellent human factors if they are to be effectively used by people throughout their activities. Personal computing will provide the quality of computing resources needed to support the human factors.

Just as languages are not a sufficient goal, environments to support their use are not sufficient either. Systems to support system development (or for that matter the NSA Analyst) are only as good as the people that use them. After all, the systems are only mechanical embodiments of some things which the Users understand. It is still up to the Users to have the intelligence to take the step beyond!

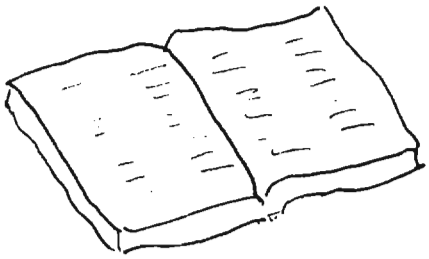
Conclusion

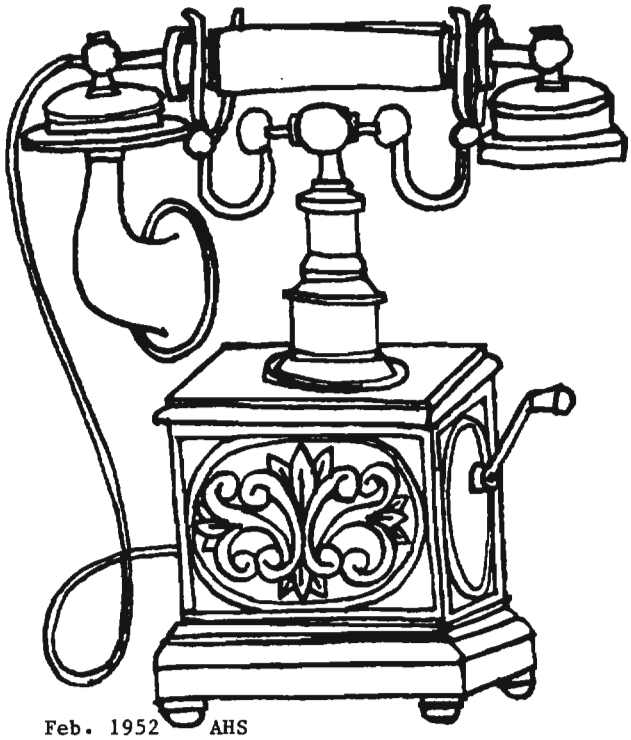
The development of systems has been explored with respect to the pressures on Agency Analysts, rapid advance of available computing technology, rising expectations of the User community, limited capacity to produce new systems, and increasing difficulty of formulating meaningful requirements. Among the various approaches that will be used to deal with future analytic problems will be an increasing use of interactive computing systems to provide advanced environments. Human factors requirements are among the most difficult and most critical aspects of these future systems. Prototyping has been suggested as a valuable experimental component in the requirement and design process.

> *the use of a prototype system*
> *can reveal important insights*
> *that are not apparent*
> *from a requirements document*

Old Phone Books Never Die (U)

by William M. Nolte, T54





In addition to sundry other forms of aging paper, the History and Publications Division (T542) maintains a file of telephone directories prepared by NSA and predecessor organizations. This file, when used in conjunction with old organization charts and other documents, has been a valuable resource in preparing books and articles produced as part of the Cryptologic History Program. The directories are also helpful in answering the numerous reference requests handled by this office. They supplement our understanding of changes in Agency organization and frequently provide the most readily available source for the correct spelling of an individual's name or the organization to which an employee was assigned at a given time.

Directories currently on file include:

Date	Station or Organization
Jan. 1946	Vint Hill Farms Station
Sep. 1948	Arlington Hall Station (AHS)
July 1949	Naval Communications Station
Oct. 1950	Naval Security Station
Apr. 1951	Naval Security Station
June 1951	AHS
Dec. 1951	Naval Security Station
Feb. 1952	AHS
Oct. 1952	AHS
Oct. 1952	Naval Security Station
Apr. 1953	NSA
Dec. 1953	NSA
Jan. 1954	AHS
Oct. 1954	NSA
Oct. 1954	AHS
May 1955	AHS
June 1955	NSA
1956	NSA (Organizational Supplement)
June 1956	NSA
July 1956	NSA (PROD interim listing)
Fall 1956	AHS
Nov. 1956	NSA (Organizational)
Mar. 1957	NSA (Organizational)
Spring 1957	AHS
Fall 1957	AHS
Oct. 1957	NSA ("New Site")
Mar. 1958	NSA (Organizational)
May 1958	NSA
Oct. 1958	NSA
Jan. 1959	NSA (Organizational)
Sep. 1959	NSA
Jan. 1960	NSA (Organizational)
Jan. 1961	NSA
Oct. 1961	AHS
Mar. 1962	NSA
Mar. 1963	NSA
Mar. 1964	NSA
June 1964	NSA
July 1965	NSA
Sep. 1965	AHS

July 1966	NSA
Apr. 1967	NSA
May 1969	NSA
July 1971	NSA
Jan. 1972	NSA
Jan. 1973	NSA
Jan. 1974	NSA
Jan. 1975	NSA
Jan. 1976	NSA
Jan. 1977	NSA
Jan. 1978	NSA
Sep. 1978	NSA
May 1979	NSA
Apr. 1980	NSA
Nov. 1981	NSA

Unless described as organizational in format and organization, all the directories in the file are arranged alphabetically by employee name. Some of the organizational directories list all or most of the individuals assigned to a particular element, but most list only the chiefs or other principals.

~~(FOUO)~~ Though useful for many reference purposes, the directories are not infallible. The move to Fort Meade and one or two major reorganizations must have been confusing and stressful for everyone involved, but the persons responsible for the telephone system must have been especially pleased to see the dust settle after such reshufflings. One practice of interest, long since abandoned, was the use of directories to reinforce security consciousness. Dividing each letter of some directories of the late 1950s are security slogans of the sort seen on posters and other devices. "Dignity of man can be shattered by a careless tongue," cautions one such reminder in the May 1958 listing. The same edition also warns that "A secret's a secret only as long as its kept," a truism that appears only a few pages after the listings for William H. Martin and Bernon F. Mitchell (both of REMP 13 on 5323 secure, 7147 outside).

~~(FOUO)~~ Agency personnel holding outdated directories not listed above may wish to consider forwarding them to T542 for inclusion in the file. Such action entitles the donor to membership in the Pack Rat Society and the gratitude of the staff. By the same token, this office also would like to acquire Baltimore and Washington area "white page" directories from the 1940s through the late 1970s. We do <u>not</u> need every year within that span, nor do we need 247 copies of the 1979 editions. Persons with directories they may be willing to part with can call 2355s before putting them in the mail.

P.L. 86-36

HUMAN FACTORS CORNER

Consumer vs. Computer

BY [redacted] P13

REVIEW: "Consumer Difficulties with Computerized Transactions: An Empirical Investigation," T.D. Sterling, Communications of the ACM, Vol. 22, No. 5, May 1979, pp. 283-289.

Reprinted from Human Factors Letter 1-80, published by CISI Human Factors SIG.

If you are among the overwhelming majority of citizens who have wondered again and again why "computers make so many mistakes", this article has much to offer in the way of enlightenment, if not in hope for improvement. If you share my suspicion that many of the goofups in bills and mailings are unnecessary consequences of bad design and poor management, Dr. Sterling's paper will provide confirmation. But the real surprise (to me, at least) is the strong indication that a considerable portion of the snafus are a consequence of deliberate and questionably honest business practices aimed at putting, and keeping, the customer at a disadvantage! The answer to "why computers make mistakes" is only sometimes "because programmers or data-entry clerks or system designers make mistakes." In some proportion of cases, the answer may well be "because computers were deliberately programmed to make those 'mistakes'."

Noting that "errors in computer produced bills and various communications sent by government (and other) agencies or businesses to citizens, clients, customers, and consumers are both commonplace and embarrassing to the computing industry ...", Sterling asks the following questions: "How many and what types of errors occur? What is the reaction of consumers after they encounter errors?" Few, if any studies have been made to answer questions like these (a fact which, in itself, seems strange to me). In 1977, a study was made by an organization or individual that Sterling calls the "Computer Ombudsman of Vancouver", in cooperation with the Consumer's Association of Canada (B.C.). This paper discusses the study's findings. (From the point of view of data gathering methods in human factors work, this type of study comes under the general heading of "critical incident" studies: focussing on incidents where a human-machine

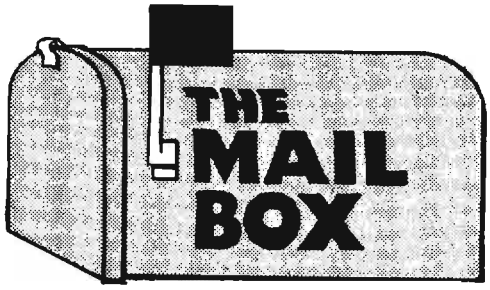
system broke down, malfunctioned, or occasioned some class of events or outcomes of interest in evaluating its strengths and weaknesses.)

A questionnaire was mailed to a random sample of members of the Consumer's Association, asking for experiences of household members with computerized transactions. In-depth telephone interviews were then conducted with people who reported specific problems in the questionnaire. The population was, according to Sterling, representative of "a broad range of middle-income groups". He frankly admits that it was probably rather atypical in one respect, namely the consumer orientation and the consequent aggressiveness and willingness to do battle in getting errors corrected. It was found that 59.5% of the households reported no problems with computerized transactions, but 25.1% had one, 10.4% had two, 3.6% had three, and an especially unlucky 1.4% had four such problems during the preceding twelve months. By far the largest proportion of the errors were related to department stores, credit cards/services, utility companies, and mail order business. Billing errors accounted for 81.2%, while all "dealing with governments" accounted for only a tiny 2.4% of the goofups.

When we see what the study showed about the different types of billing errors, a rather suspicious pattern seems to become even clearer. Almost all the errors result in a net financial gain for the billing entity: they include such things as charges for nonexistent expenditures, inappropriate charges of interest, and overcharges. Sterling points out that errors in the consumer's favor may well have been under-reported, but the balance is overwhelmingly in the other direction. In 73% of cases, respondents succeeded in getting the error corrected. Whether success attended their efforts or not, attempts to get errors corrected required one contact in 36.6% of cases, up to three contacts in 30.4%, and *four or more* contacts in an appalling 33% of the cases! It seems evident that both victims and offenders were admirably persistent and determined. Sterling points out that "each contact between a client and an organization requires additional 'overhead' time. The number of contacts is [a] measure of effort." The average time from discovery of an error to its settlement was 8 weeks, but 15% dragged on for 20 weeks. Consumers spent an average of 2.6 hours of their time in the attempt, but 20 hours or more were expended in 20% of the cases. In 10% of cases, it took extra effort to remove incorrect interest charged on top of incorrect charges, and in 2% of cases it was never removed. Trying to correct an error was often made unpleasant for the consumer. 16% of respondents reported having been coerced in some way to pay a disputed bill, often with an implied threat of damage to their credit rating. 8% were "treated as troublemakers".

Some customers, while continuing to deal with certain businesses that frequently offended, elected to pay cash in order to avoid the hassles, or to buy a magazine at a news stand rather than maintain a subscription. While Sterling does not explicitly apportion blame for the error-prone practices, he presents important evidence that points toward deliberate management philosophies as well as bad design of computing packages. Another study made in 1976 showed a 7-day delay on the average between the billing date and the time the bill was mailed, with a range of from 3 to 20 days delay! In the 1977 study, most of the victims of incorrect interest charges were also victims of this late-mailing practice. In addition, it is clear that "in almost 60% of the cases where interest had been charged on some disputed amount, it was *not* removed at the time when the charge turned out to be incorrect." This must be a consequence either of bad program design *or intentionally poor service.*

Sterling offers the following in summary: "We might ... ask if, with the use of all the power computers add to the task of management, it is really necessary to have systems that fail to correct errors, fail to inform customers where they may address a complaint, fail to remove interest in cases of billing errors, fail to adjust the printing of bills to the capacity of an organization to mail them out, and fail to do all the other things which careful examination of existing packages seems to uncover." He expresses a valid concern about the trend in the business community toward electronic fund transfer and a "cashless economy". "What will happen when the visible audit trail, such as checks, bills, and receipts, is replaced by electronic signals?" Next time you get an erroneous bill in the mail, don't blame the computer, the programmers, or the data entry clerks. Instead, consider placing the blame on business and management practices that encourage the design of bad and dishonest commercial software packages!



Dear CRYPTOLOG Editor,

(U) I am shocked that you would print a personal note from me to you in CRYPTOLOG (see Letters to the Editor, CRYPTOLOG, January 1982) without my knowledge or consent. As a result of your action a person of U/I sex ran into my office, bared its chest and demanded to know if it won the "Strangest Bust of the Month" contest, and "what was the prize?"

(U) A bust is not only an arrest when you are caught with the goods, a marble statue of George Washington, etc., but it is also a cryptographic abnormality (e.g. an error in encryption, a fault in equipment, a violation of rules for use) that may lead to some analytic understanding or exploitation of a cipher system.

publication.

(U) Sorry, no prizes.

[redacted]
B63 x5311

P.L. 86-36

I think it was [redacted] who said, on more than one occasion, "Whatever I get, I publish." Or maybe it was Art Salemme.

Ed.

Dear Wayne,

(U) A gentleman in one of the operational groups has correctly pointed out that Mark Twain died in 1910, and not 1909 as I had indicated in my article "The Literary Bends" (January's CRYPTOLOG). I respect his wish to remain anonymous. But to set the record straight, and maybe to do a little more drumming and trumpeting, could you publish my answer to his letter?

"Dear (Sir),

You're right! Sam died in 1910. Unfortunately, I implicitly trusted John O'Hayre's statement on page 106 of his Gobbledygook Has Gotta Go (I referenced his book in the footnote to the article). He says it was 1909.

What to do? I guess not much can be done, other than to express my sentiments like so:

Shall I point at O'Hayre
and say he is the one,
Who caused the mistake
I should never have done?
No. I'll just throw up my hands,
in spite of the shame,
And cite that cruel law
that uses my name.

I very much appreciated your comments.

Sincerely,
Al Murphy (E41)"

Thanks, Wayne. AM

A Toy Problem (U)

by David J. Tiren, G9

P.L. 86-36

(U) Determine the callsign system being used:

Stations:	1	2	3	4	5	6	7	8
Day 1	DRQ	YNM	KPH	ZLH	TIH	HWZ	QIM	OWK
Day 2	FGT	QBP	ESF	ZOE	UAE	ETI	TAP	RTN
Day 3	LJI	SFD	BHV	RCV	XWV	VUA	IWD	GUB
Day 4	CON	VKJ	AMY	UGY	WZY	YXW	NZJ	JXF

(U) Answer next month (the *original* answer from Dave came to us in a plastic bag, if that's any help to you!).

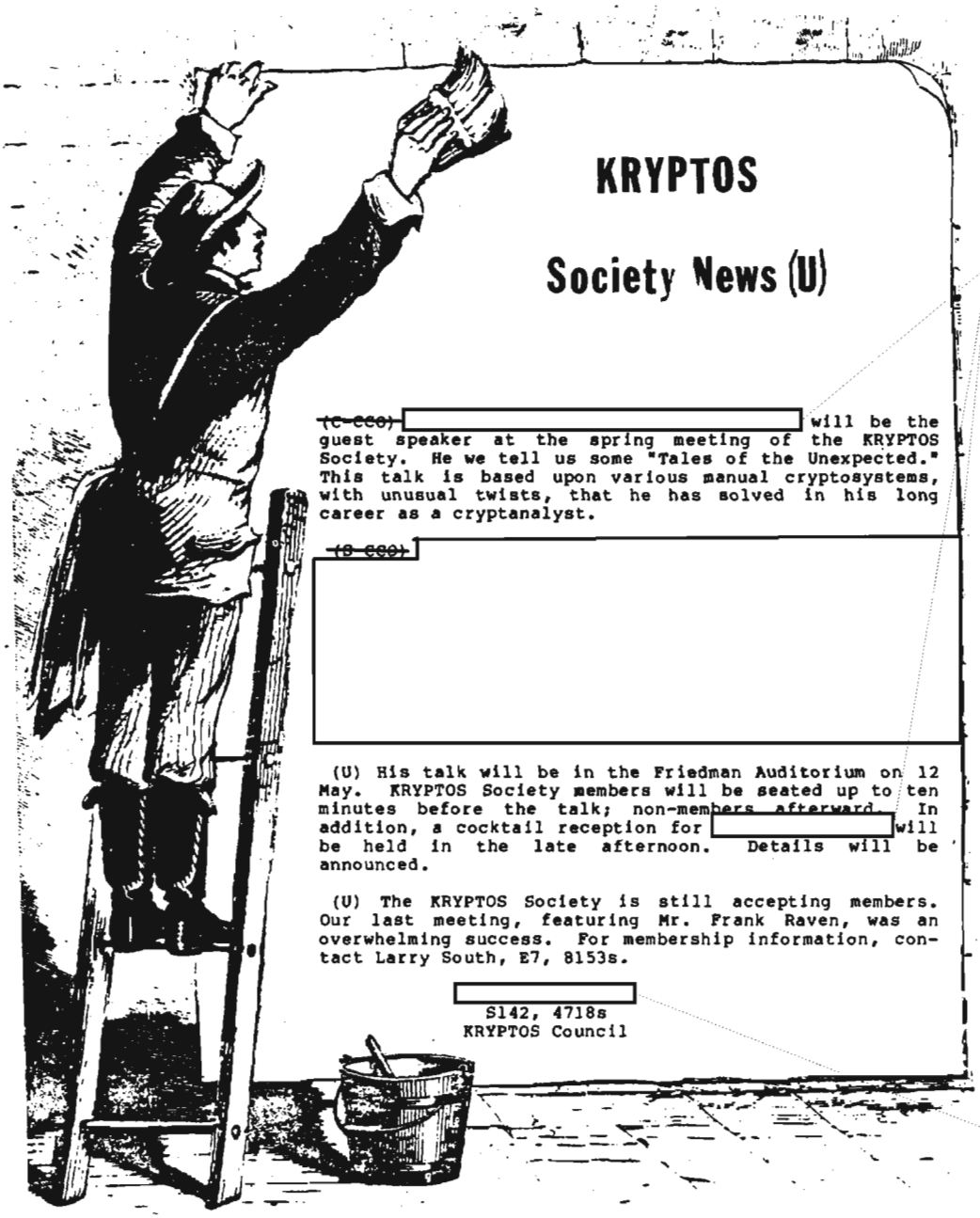
SOLUTION TO NSA-CROSTIC No. 38

"[Some] Thoughts on Lexicography," S[tuart H.] Buck, CRYPTOLOG, September, 1974.

"Glancing through the latest bulletin of the Mongolia Society, my eye [came upon] the following remark by John Krueger, professor of Altaic Studies at Indiana University:
'The very worst possible way to make a dictionary, and...the way that nearly all appear to be made, is to make a grand compilation of all existing dictionaries...'"

For those who are interested, here is the remainder of the quotation:

"...possibly abridging slightly and adding a few examples. The obvious and ideal way, seldom followed, is to begin with a set of texts, draw from them only the words used in those texts, and create a dictionary out of the actual recorded usage of the literature (or in the case of a spoken dialect, from the noted speech of the speakers). The latter and better course is self-evidently vastly more difficult and time-consuming."



KRYPTOS

Society News (U)

EO 1.4.(d)
P.L. 86-36

~~(C-CCO)~~ [redacted] will be the guest speaker at the spring meeting of the KRYPTOS Society. He we tell us some "Tales of the Unexpected." This talk is based upon various manual cryptosystems, with unusual twists, that he has solved in his long career as a cryptanalyst.

~~(S-CCO)~~ [redacted]

(U) His talk will be in the Friedman Auditorium on 12 May. KRYPTOS Society members will be seated up to ten minutes before the talk; non-members afterward. In addition, a cocktail reception for [redacted] will be held in the late afternoon. Details will be announced.

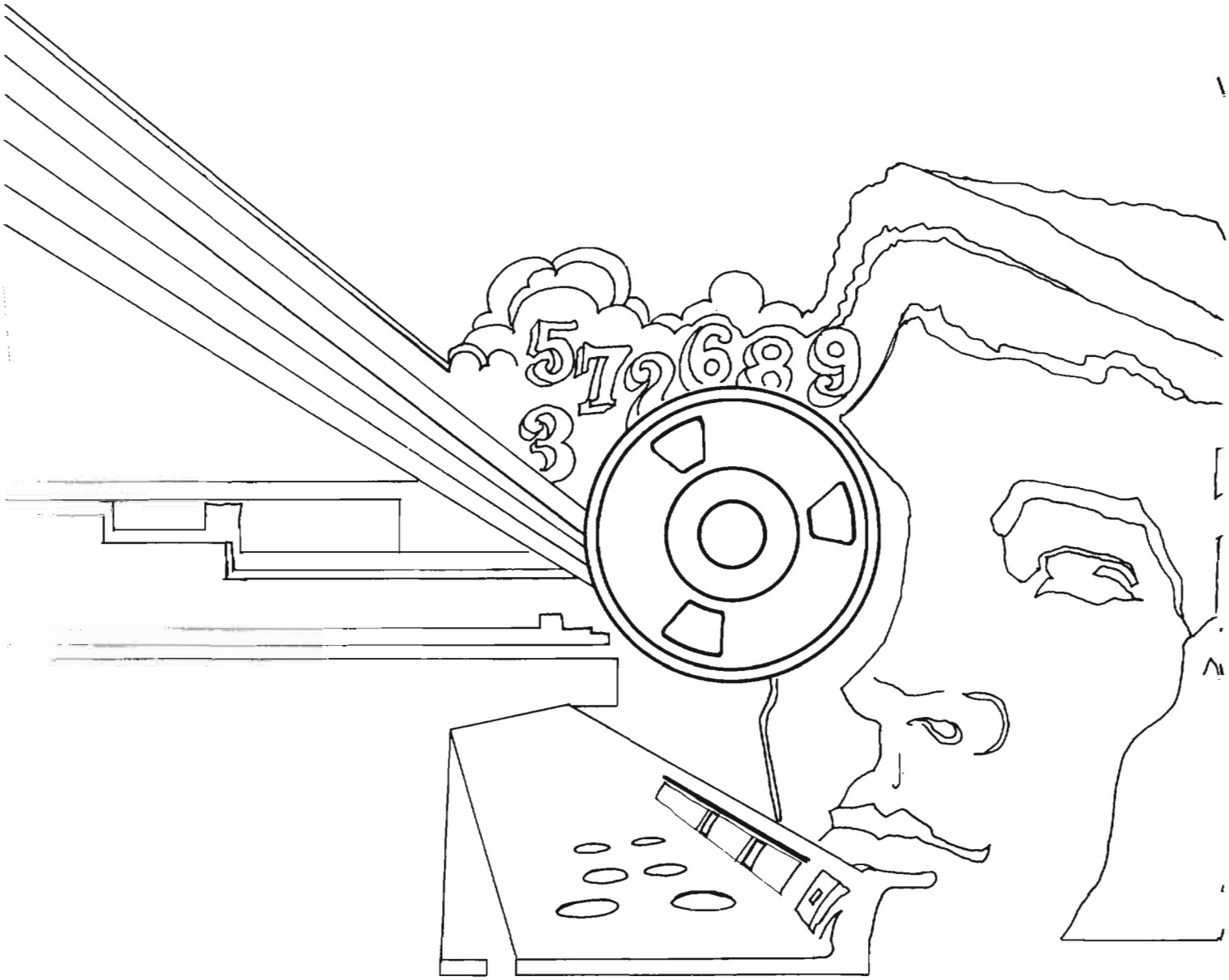
(U) The KRYPTOS Society is still accepting members. Our last meeting, featuring Mr. Frank Raven, was an overwhelming success. For membership information, contact Larry South, E7, 8153s.

[redacted]
S142, 4718s
KRYPTOS Council

P.L. 86-36

FOR RENT

Reasonable


5 7 9 6 8 9
3