NOT FOR QUOTATION
WITHOUT PERMISSION
OF THE AUTHOR

USING THE COMPUTER TO COMMUNICATE: A USER'S GUIDE TO *TELECENTER*

Michael M. L. Pearson
Carolyn L. Lathrop

WP-80-109
June 1980

Working Papers are interim reports on work of the International Institute for Applied Systems Analysis and have received only limited review. Views or opinions expressed herein do not necessarily represent those of the Institute or of its National Member Organizations.

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

PREFACE

The Survey Project is exploring the usefulness of computerized conferencing as a craft tool for applied systems analysis. In cooperation with IIASA's Computer Services Department it is working to 1) develop effective procedures and practices, 2) produce useful introductory documentation, and 3) facilitate computerized conferencing for IIASA scientists as part of a teleconferencing dissemination/evaluation research activity.

For readers unfamiliar with IIASA and the Survey Project we offer the following two paragraphs for a better understanding of the context in which we are exploring computerized conferencing. It is this context: IIASA's research and its researchers, that gives relevance to "using the computer to communicate."

The **International Institute for Applied Systems Analysis (IIASA)** is an interdisciplinary, nongovernmental research institute, chartered in October, 1972 upon the initiative of the academies of science or equivalent institutions of twelve nations (there are now seventeen National Member Organizations). By applying systems analysis, its staff of scientists from East and West is seeking to gain a better understanding of important contemporary problems resulting from scientific and technological development. IIASA conducts much of its research in cooperation with other research and policymaking organizations worldwide.

The IIASA **Survey Project**–a project to survey the state of the art of applied systems analysis–was established to promote the development of applied systems analysis and to disseminate its methods and approaches. The Project seeks to encourage the widespread and better application of systems analysis to problems of international relevance; to improve analytical techniques and their usefulness to decision processes; to contribute to the education in systems analysis of the expert and the interested nonexpert. To pursue these objectives it is: 1) publishing a series of books on applied systems analysis, 2) writ-

ing a *Handbook of Applied Systems Analysis* in three volumes, and 3) conducting research into the craft of applied systems analysis.

This paper was printed using a Varian electrostatic line printer driven by IIASA's PDP-11/70 running under the UNIX operating system.

CONTENTS

USING THE COMPUTER TO COMMUNICATE: A USER'S GUIDE TO *TELECENTER*

Michael M. L. Pearson and Carolyn L. Lathrop

INTRODUCTION

This working paper provides a brief description of how to use *telecenter*, a rudimentary in-house computerized conferencing system. Although *telecenter* is designed for people with no previous computer experience, this guide presumes that the reader is already somewhat familiar with IIASA's editor programs.*

Telecenter, or *tc* as we shall call it here for short, is a collection of UNIX "C-shell" command interpreter programs assembled by the Survey Project with assistance from Computer Services. It was designed to facilitate wider participatation in the Survey Project's current telconferencing dissemination/evaluation activities and to explore computer conferencing as a craft tool of systems analysis. At the moment conferences on *tc* are established by the Survey Project. There are no protection mechanisms to ensure privacy or to guard against loss of text in the event of system failure.

We hope that *tc* will be the forerunner of more advanced IIASA-based telecommunications/office automation software to which all institute staff will someday have easy access. Certainly if present trends initiated by IIASA's Computer Services Department continue, this will be the case. We believe that *tc*-type software represents the central component in a complex of software needed to revolutionize conventional ways of organizing, storing, retrieving and transmitting information for routine administrative and research activities at IIASA. We refer the reader to (Pearson, 1980) for more details on this subject, and, in particular, to *The Office of the Future* (Uhlig, Farber, Bair) for a general introduction to the recent trends in micro electronics, text

*For anyone desiring more detailed information on the *edx* screen-editor program embedded in *tc*, we refer him/her to *Using the Computer to Communicate: An Introduction to Text Processing at IIASA* (forthcoming working paper in this series.)

processing, distributed computing, and networks as they relate to computer messaging and office support systems.

This working paper is the second in a Survey Project series on using the computer to communicate. It is primarily a tutorial--a cookbook--on how to use *tc's* basic features. This series is an attempt to collect in one place the basic information a non-computer-expert at IIASA will need to use the computer to communicate. As mentioned before, we are starting with basics.

We caution the reader that *tc* in its present form is doomed. Its implementation is inadaquate, it has almost no privacy or protection features, and, perhaps even more importantly, its programs must be re-written in a more efficient code. It is a crude first attempt to determine the most appropriate teleconferencing features for IIASA users. It is a prototype that, we hope, will be replaced by better versions. In fact, it is with some reticence that we talk about *tc* in a working paper such as this one. By doing so we run the risk of prematurely freezing computerized conferencing development work--something that we feel would be a great disservice. It is vital to preserve flexiblity in design and move towards better functions within a real-world, applications framework.

Even with its many inadequacies and inefficiencies *tc* in its short life has taught us a number of important lessons. We found that:

- IIASA needs a mechanism for locally distributing those computer-mediated messages and conference comments received from systems outside the institute.

  We have experimented with *tc* as a vehicle for making other computerized conferencing systems transparent to IIASA users.* With such a scheme a local user needs to learn only one system in order to access a variety of other systems--each with its own procedures and conventions. Also, a message from outside the institute needs to be transmitted to IIASA only once for subsequent multiple distribution--a savings in communications charges, especially when messages have large texts and the character transmission costs are high (see Pearson, 80.)

- UNIX(TM) is particularly suited for developing *tc*-type software.

  The simplicity and power of IIASA's UNIX operating system constitutes a unique tool for the development of communications software of the *tc*- and office-automation-type. This is especially true of that version of UNIX currently running at IIASA. It incorporates the powerful *C-shell* command interpreter language and contains highly useful interactive features for stopping and starting programs and moving quickly through deep directory heirarchies. These UNIX-based tools, largely IIASA-specific, were

---

*For example, one current *tc* conference concerns a forthcoming IIASA book. The authors, one at IIASA and the other in Wisconsin, have been using EIES -- the Electronic Information Exchange System of the New Jersey Institute of Technology -- to communicate (see Hiltz, Turoff 1978). The latest member of the *tc* conference, the IIASA editor responsible for copy editing the manuscript, need know nothing about the New Jersey system in order to join in discussions regarding the manuscript. In addition, all prior discussions between the joint authors are now available to the new local participant.

developed and implementd by Computer Services as part of a general policy of advanced system development. They may be of little interest to individuals desiring faster and bigger FORTRAN programs or larger "number-crunching" modeling programs. They are, however, the foundation for innovative developments in "using the computer to communicate" in the sense used in this series.

To give an example of what we mean, it took approximately 20-25 man/hours, using *C-shell* command interpreter language, to create an operational *tc*. This was due largely to the availability of UNIX's file/directory hierarchy as a mechanism for structuring information and to the power of existing UNIX utility programs. The development work was an excellent example of building on the labor of others and avoiding the need to "re-invent the wheel" at every turn.

Also, thanks to UNIX's simplicity, individuals with very little experience in programming are able to modify *tc's* functions to user needs. Later it will be comparatively easy to "tool up" the *tc* functions we find desirable by converting from *C-shell* code to that of a true programming language such as *C*. First, however, we must learn more about the basic functions needed by IIASA's user community.

- Short response time is critical for the success of *tc*-like tools.

IIASA's PDP 11/70 computer is currently plagued by response time problems due, in part, to the large number of terminals now connected to it. This has been the case for some time. As far back as the end of 1979 Computer Services recognized that "Recently, as demand for computing services has exceeded our capacity, users have been disturbed by the lack of good response time and various other bottenecks." (Kulp, 1979, p. 7) and pointed out that "... most such problems are already being solved (equipment ordered, software under development, etc.) ..." And, indeed, at that time major steps had been taken. Perhaps most importantly, order had been taken on a powerful new main CPU--a VAX 11/780—to augment the existing PDP 11/70. And it should be noted that at this writing delivery has been taken on the VAX and relief for IIASA's response-time problems seems just around the corner.

We make this small digression into recent IIASA computing history because institute response-time problems have coincided with our efforts to explore the potential of *tc* for IIASA users. As a result we wish to stress the importance of short response time for tools such as *tc*. Without it, such computer tools are meaningless. In a broader sense, the response-time issue is subsumed by the fact that,

availability

reliability

integrity

must underlie any generally available teleconferencing/office-automation services provided at IIASA.

For true availability there should be a terminal in every office coupled with short response time. A terminal that reacts to a user's command one or two minutes after the command has been entered is not truly available.* Once response time is improved at IIASA there must be a conscious institute policy to preserve it. A measure of response time should be established and it must not be permitted to drop below a given threshold.

STARTING

To use *tc* you must first find an available terminal. Most computer terminals at IIASA are connected to the in-house computer system. There are two basic types of terminal: video (sometimes called CRT) and hardcopy (typewriter). Video terminals resemble television sets, whereas hardcopy terminals look more like electric typewriters. Hardcopy terminals print on paper. Video terminals display on a TV-like screen and have a small square of light called a cursor that moves across the screen and indicates where you are on the screen while typing. When following the examples in this guide it is best to use a video terminal.

After you have found a free terminal you can tell that it is ready to use if the last line (bottom-most line that the terminal has printed) reads:

```
login:
```

A note about the conventions used in this guide: In this and the following examples, bold face type (**like this**) indicates lines the computer prints; normal type face indicates what the user types. The symbol "⏎" stands for a carriage return, marked on some terminals as "return" and on others as "CR" or "newline".

To begin, type the word "telectr" in lowercase letters, followed by a carriage return. In the example below, the user Smith types "telectr" to start his/her session:

```
login: telectr ⏎
Welcome
Name?
```

If you make a mistake when typing in "telectr" or any other command you can correct the mistake by using the delete key as long as you catch the error before beginning a new line. On most IIASA terminals this delete key is marked "del" and must be depressed while also depressing the "shift" key. On some terminals the "delete" key is called the rubout key and is marked "rub" or "rubout". When the delete key is depressed the cursor moves to the left and the unwanted character disappears. You can depress "delete" as many times as you need to. Depressing "delete" and/or "rubout" once the cursor has backed up to the left-hand margin will simply cause the terminal to beep harmlessly at

*By analogy, consider the frustration that would result from having to pause for up to several minutes each time you reached for something on your desk.

you. On some terminals, depressing the delete key longer than a second or two has the same effect as repeatedly striking the key rapidly.

If there is a "message of the day" from Computer Services it will print out on the terminal before the word "Welcome" appears.

Next, Smith types "smith" (again in lowercase letters) in response to the query "Name?". *Tc* responds by printing out an overview of all the conferences in which Smith is a member and how many conference comments he or she has not yet read:

```
login: telectr ⏎
Welcome
Name? smith ⏎
Conf: eies 3 comments. smith up to date
Conf. nroff: 16 comments. smith has 2 outstanding

To see new items type "seecomments <conference-name>"
To enter new item type "newcomment <conference-name>"
To modify existing item type "modify <conference-name> <comment-number>"
To have an overview of all your conferences type "overview"
To see status of conf. participants type "status <conference-name>"
To see this list type "help"
To quit type "--"

**
```

In this case Smith is a member of two conferences. One is called "eies" and contains a total of 3 comments; the other conference, called "nroff", contains 16 comments. Smith is "up to date" in the eies conference, meaning that s/he has read all three comments that have been entered so far. In the "nroff" conference, however, there are two comments that Smith has yet to see.

After printing out this conference overview, *tc* displays a list of frequently used "commands" from which Smith may choose. Smith can choose to: see waiting comments, make new comments, modify existing comments, obtain a new list of available commands, see the status of other conference participants, or to quit *tc*. *Tc* offers more commands than those that appear in this initial list; these will be discussed later in this docment.

LEARNING THE STATUS OF A CONFERENCE

Tc is now waiting for input, that is, waiting for Smith to tell it what to do. The characters "**" are called a "prompt". Their appearance means that Smith may now type a command to tell *tc* what to do. If Smith types something *tc* doesn't understand, *tc* will respond "Command not found." and print the prompt "**" again. Typing a carriage return in response to the prompt simply causes the prompt to appear again.

In the example below, Smith wishes to find out the "status" of the participants in the "nroff" conference; that is, s/he wants to know if the other participants have read all available comments. Referring to the command list printed out above, Smith types "status nroff" (without the "<" or ">" surrounding the conference name) followed by a car-

riage return:

```
** status nroff ⏎
CONFERENCE: nroff
UP TO (NOT INCLUDING) 16 (1 unread) lathrop
UP TO (NOT INCLUDING) 15 (2 unread) smith
UP TO DATE            mpearson

16 ITEMS

**
```

The statement beginning "Up to (not including) 16" indicates that the participant Lathrop has read 15 comments and still has one waiting to be read. Thus, typing "status nroff" informs Smith that of the 16 total comments entered to date, M.Pearson has read all 16, there is one message that Lathrop has not yet read, and Smith still has two waiting to be seen.

SEEING CONFERENCE COMMENTS ENTERED BY OTHERS

Smith now wants to read the two pending "nroff" comments. New comments can be read by typing the command "seecomments" followed by the name of the conference and a carriage return. Below, Smith uses "seecomments" to see the the two unread comments waiting in "nroff".

```
** seecomments nroff ⏎

Seecomment: Depress carriage return to see next item. Terminate with "--".
```

"Seecomments" first responds with an instruction. To read the first unread conference comment, Smith needs only to press a carriage return; *tc* will display that comment on the screen, one screen-full at a time. If the comment is lengthy and takes up more than one screen (more than 24 lines), the next screen-full of text can be read by again pressing a carriage return. On days when the computing system is heavily loaded, *tc* can be slow to respond. It is tempting to press several carriage returns in an attempt to speed things up, but patience pays off in the long run. It is important not to get ahead of the "seecomments" command by pressing additional carriage returns, for when *tc* finally does respond, the awaited comments will zip by on the screen at a rate that is too rapid to read.

"Seecomments" prompts with this same "Depress carriage return" instruction after each message presented. If, for some reason, Smith does not want to read the next comment, s/he can terminate the "seecomment" command by typing "--", and will receive a ** prompt. In this case, though, Smith does want to read the next comment, and responds by pressing a carriage return:

```
e

Conf: nroff              Comment: 15          Lines: 2
Entered by: mpearson    on: Tue Apr 1 14:42:27 CET 1980
Associated IIASA Comment: 14
Keys: WP ready

The draft of my working paper is finished.  I would appreciate
any comments you may have on it.

Seecomment: Depress carriage return to see next item. Terminate with "--".
```

Each comment entered in a conference in *tc* has a header indicating the conference it belongs to, its own serial number in that conference, the number of lines of text it contains, its author, date/time of submission, an "associated IIASA comment" (the number of another comment in the conference to which it refers) and a line of "keys", that is, words summarizing in brief the contents of the message. The text of the comment follows the header.

To read the next comment, Smith again depresses the carriage return:

```
Seecomment: Depress carriage return to see next item. Terminate with "--".
e

Conf: nroff              Comment: 16          Lines: 4
Entered by: mpearson    on: Tue May 6 08:57:04 CET 1980
Keys: varian program available

How are things coming on your text editing tutorial?
I have written a short program to simplify sending text
over to the varian.  You can try it the next time you
want to print out your tutorial.
**
```

Note that after all pending messages have been read, *tc* again responds with the ** prompt.

Using "seecomments" brought Smith up to date. Now checking the status of "nroff" indicates that Smith has read everything.

```
** status nroff
CONFERENCE: nroff
UP TO (NOT INCLUDING) 16 (1 unread) lathrop
UP TO DATE            smith
UP TO DATE            mpearson

16 ITEMS

**
```

ENTERING YOUR OWN COMMENT IN A CONFERENCE

Tc provides each user with a convenient means for composing and modifying the text for new conference comments. The command used to enter a new comment is "newcomment" followed by the name of the conference and a carriage return. Below, Smith uses "newcomment" to enter a comment in the conference "nroff":

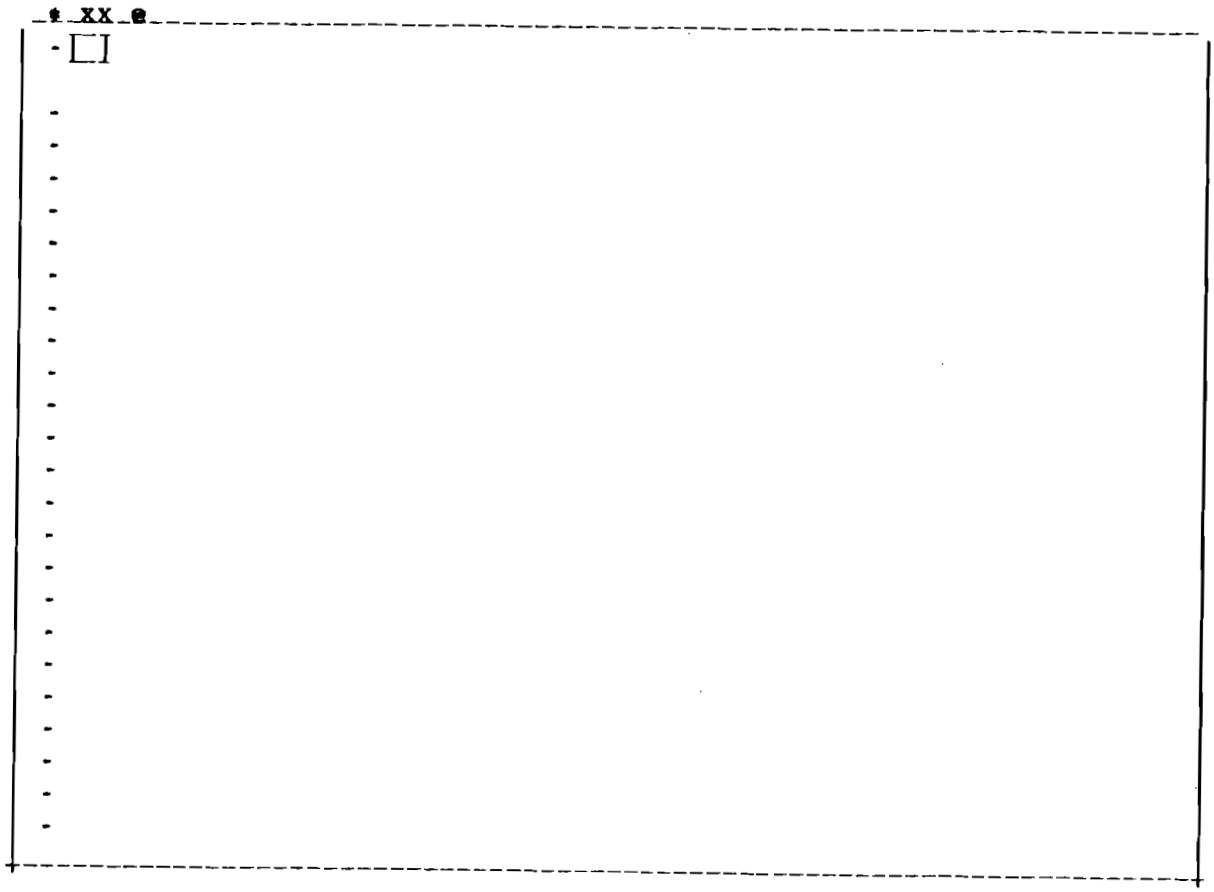
```
** newcomment nroff e
(You are in the IIASA editor. Type "x" to use edx.
Type "w" then "q" to enter comment.)
ENTERING SCRATCHPAD:
0
*
```

The command "newcomment nroff" puts Smith in a work area called "SCRATCHPAD" used for composing the new conference comment. In addition, *tc* automatically supplies Smith with IIASA's standard editor program. A "0" prints out because the editor program is reporting that SCRATCHPAD currently has zero characters in it. The "*" is the editor program's prompt. *Tc* also reminds Smith to type a "w" and then a "q" to quit the IIASA editor (this will be discussed in more detail later).

All composition of text on *tc* is done in SCRATCHPAD. Each user has his or her own SCRATCHPAD; this one SCRATCHPAD is used to compose text for all conferences in which s/he is a member. As we will see below, *tc* asks at certain points whether or not to delete SCRATCHPAD. It is perfectly acceptable to leave written material in SCRATCHPAD, to quit *tc* and then come back another time for further additions or modifications to the text. Had Smith's SCRATCHPAD not been empty, some number other than "0" would have been printed--the number of characters in SCRATCHPAD, and Smith would have found the material written earlier.

Since Smith is in the standard IIASA editor program he or she can type "x" or "xx" to go into screen display editor "edx".

In the example below, after the editor's prompt "*" appears on the screen Smith types "xx" and a carriage return. This starts "edx"'s full-screen display. After "xx e" is typed, anything appearing on the screen is erased and the entire screen of the terminal looks something like the display enclosed in a box in the example below.



The screen's left-hand margin becomes a column of hyphen characters and the cursor appears at the upper left-hand corner. (In our edx examples we will represent the cursor with the symbol [] At this point just typing text and doing carriage returns is all it takes for Smith to write the text of the new comment.

Below we see the results of Smith having carefully typed three error-free lines in SCRATCHPAD.

```
|Let's think about making a macro package that anyone at
|the institute can use.  There should be one set of input
|conventions for all text-processing here.[]
-
-
-
-
-
-
-
-
-
-
```

The "delete" key may be used in "edx" to fix typing mistakes as long as the mistake is discovered and corrected before a new line is begun. By typing carefully and fixing typing errors immediately after they occur, Smith is able to enter the desired text into *tc's* SCRATCHPAD. This is the simplest and most straightforward means of using "edx" to compose text, but does not take full advantage of "edx's" text-editing powers. Most commands in "edx" are control functions, which means that they are given by depressing the "control" key and another, appropriate key at the same time. Commands in "edx" allow you easily add, subtract, re-arrange and otherwise modify text. For information on using "edx", consult the *Unix Programmer's Manual*, or the Survey Project tutorials, *An Introduction to Text Processing at IIASA*, and *Text Processing Workbook*.

In the following example, Smith finishes typing the text in SCRATCHPAD, and depresses first the escape key (marked "esc" or "altmod" on the terminal keyboard) and then "q" to leave "edx" and return to the editor. No carriage return is necessary at this point. Note that the "escape" key functions differently from the "control" key used in most editor commands. The "escape" key must be depressed *before* the "q" key, whereas "control" is always depressed *simultaneously* with another key. The notation <escape...q> is used here to indicate that the "escape" and "q" keys are typed sequentially. The words "escape q" will not actually appear on the screen as you type. <escape...q> terminates the full-screen display of edx and causes the cursor to move to the lower left-hand corner of the screen. At this point, Smith is again in the standard editor -- a fact demonstrated by the appearance of the editor prompt "*".

Next Smith types a lowercase "w" (meaning "write"), followed by a carriage return, to make a permanent copy of the contents of SCRATCHPAD on the computer. After the computer responds by printing out the number of characters in

SCRATCHPAD (in this case 155), Smith types a "q" and a carriage return to leave the SCRATCHPAD and start the sequence for entering the new comment in the conference. Remember that writing the contents of SCRATCHPAD (the "w" command) is one of the most important steps in entering a comment. If Smith quits the editor program (the "q" command) without first typing a "w", the contents of SCRATCHPAD will be lost and have to be retyped. Fortunately, the editor program reminds you to write your file by asking, ?write if you try to quit before typing a "w" command. More importantly, however, if the system "crashes" (is temporarily inoperable), any unwritten material in SCRATCHPAD will be lost. It is advisable to "write" SCRATCHPAD frequently, (say, every half page or page) so that if the system does crash unexpectedly, losses will be minimal. (As usual, "edx" may be re-entered after the "w" command by typing an "x" and a carriage return in response to the editor prompt, *).

```
|Let's think about making a macro package that anyone at
|the institute can use. There should be one set of input
|conventions for all text-processing here.
<escape...q>
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
*w @
155
*q @
```

After the "q" is typed, *tc* asks first for an associated comment number and then a key. If no associated comment and/or no key is desired, simply press a carriage return. In our example Smith gives "12" as the associated comment and gives "standard macro package for IIASA/text-processing" as the keys.

```
Associated IIASA Comment (*)? 12 e
Keys (Word/Phrase/)? standard macro package for IIASA/text-processing e
Associated Comment: 12
Keys: standard macro package for IIASA/text-processing
OK to send (y/n)? y e
```

If at this point Smith decides *not* to enter the message, s/he can type "--" in response to either the "Associated IIASA Comment?" or the "Keys?" question. This terminates the sending process, and *tc* prompts again with a ** . However, any text written in the SCRATCHPAD will still be there when Smith next uses "newcomment". This can be useful for last-minute text alterations.

Next *tc* asks if it is okay to enter the new comment. This is the last point at which Smith can reconsider sending the message. Typing "n" would abort the entire process and result again in a ** prompt. Since Smith types "y" the comment is entered. *Tc* then reports on the contents of the comment's header and, finally, asks if Smith wants to delete the SCRATCHPAD. Typing a "y" and agreeing to erase SCRATCHPAD is the most usual response to this question -- it assures that you have a clean work area for composing the next conference comment. If Smith types "n" then the text s/he just wrote will remain in the SCRATCHPAD. This can be useful for sending the same material to more than one conference. Incidently, *tc* treates a carriage return as a "yes" answer, so be careful not to inadvertantly press carriage returns -- you could send off a message without intending to!

```
OK to send (y/n)? y e
Comment being entered.
Entered as:
Conf: nroff Comment: 17 Lines: 3
Entered by: smith on: Mon May 12 08:26:46 CET 1980
OK to delete SCRATCHPAD? (y/n)?y e
**
```

ABBREVIATING WHAT YOU TYPE--GETTING AN OVERVIEW OF YOUR CONFERENCES

Now when Smith checks to see the status of the conference "nroff" we see that the participants "lathrop" and "mpearson" have an additional comment to read--comment 17 just entered by smith:

```
** st nroff @

CONFERENCE: nroff
UP TO (NOT INCLUDING) 16 (2 unread) lathrop
UP TO DATE             smith
UP TO (NOT INCLUDING) 17 (1 unread) mpearson

17 ITEMS
```

Note that Smith typed "st" in the example above instead of "status." *Tc* allows you to type any portion of a command name that is not ambiguous. This is useful to avoid typing out long commands. Typing either "st" or "sta" or "stat" or "statu" is the same as typing the full word "status". An abbreviated form of the "overview" command...

```
** ov @
Conf: eies 3 comments. lathrop up to date
Conf: nroff 17 comments. lathrop up to date
```

for example, works as well as the long form...

```
** overview @
Conf: eies 3 comments. lathrop up to date
Conf: nroff 17 comments. lathrop up to date
```

PRINTING OUT CONFERENCE COMMENTS

After reading a new conference comment, Smith may want to obtain a copy for future reference. There are several commands available on *tc* that can be used to re-read existing comments.

The "listcomment" command is the quickest way to review old comments. "Listcomment" simply retrieves a requested conference comment and displays it on the screen (or, for people using hard-copy terminals, the comment is printed out on paper). By typing "listcomment" followed by the name of the conference and the desired comment number, Smith can re-read old comments.

```
** listcomment nroff 16 e

Conf: nroff              Comment: 16          Lines: 4
Entered by: mpearson   on: Tue May 6 08:57:04 CET 1980
Keys: varian program available

How are things coming on your text editing tutorial?
I have written a short program to simplify sending text
over to the varian.  You can try it the next time you
want to print out your tutorial.
**
```

The commands "lprintcomment" and "variancomment" are similar to "listcomment", except that the requested comment is printed out on either the line printer or the varian. The syntax for these commands is identical to "listcomment": type either "lprintcomment" or "variancomment" (abbreviations also work), followed by the conference name and desired comment number:

```
**lprintcomment nroff 16 e

        (or)

**variancom nroff 16 e
```

The output can be retrieved in Computer Services, room S-30. It will be marked with a large header reading "TELECTR" and your login name.

QUITTING *TC*

It is possible to terminate your session on *tc* any time you receive a ** as a prompt. Quitting *tc* is done by typing "--" and a carriage return. The computer will respond with a "login" sign, ready for the next person to log in:

```
** -- e
login:
```

SETTING "TERSE" AND "NOTERSE" MODES; THE "HELP" COMMAND

After using *tc* several times you may become familiar enough with how it functions that you don't need to review the command list or receive a conference overview each time you log in. The command "terse" allows you to immediately receive a "**" prompt after logging in, without reviewing either the command list or an overview of your conferences first. The "terse" command may be entered any time you receive a ** as a prompt, and will affect each future login onto *tc*. In the following example, Smith sets the "terse" mode by typing "terse", then exits from *tc*. On the following login, Smith immediately receives a **, and is ready to begin a new session.

```
login: telectr e
Welcome
Name? smith e
Conf: eies 3 comments. smith up to date
Conf: nroff 17 comments. smith up to date

To see new items type "seecomments <conference-name>"
To enter new item type "newcomment <conference-name>"
To modify existing item type "modify <conference-name> <comment-number>"
To have an overview of all your conferences type "overview"
To see status of conf. participants type "status <conference-name>"
To see this list type "help"
To quit type "--"

** terse e
** -- e
login: telectr e
Welcome
Name? smith e
**
```

Of course, there are ways of easily obtaining information omitted as a result of the "terse" mode. The "overview" command, or an abbreviation of it, prints out the current state of all conferences of which you are a member. In addition, the *tc* command called "help" provides a list of frequently-used commands. Whenever you receive a ** prompt you can type "help" for a quick review of what is available:

```
** help e
To see new items type "seecomments <conference-name>".
  (Depress "newline" or "carriage return" key to see next comment or
  next page of a large comment. ** indicates that you are up to date.)
To enter new item type "newcomment <conference-name>".
To modify existing item type "modify  <conference-name> <comment-number>".
To have an overview of all your conferences type "overview".
To see status of conf. participants type "status <conference-name>".
To set terse mode type "terse". To set verbose mode type "noterse".
To quit type "--"

**
```

If you change your mind, and again want *tc* to print out a command list and conference overview each time you log on, enter the "noterse" command when you receive a ** prompt. "Noterse" negates the effect of the "terse" command. Like "terse", it needs to be entered only once to affect all future logins (or until the "terse" command is given again).

WARNINGS

For those people very familiar with the "editor" program: do not use the editor "f" command to change the name of the file in which you are composing conference comments. This could result in loss of text.

For all users: If at anytime while using *tc* the computer prints the words "try again" or "no more processes", if you should get a "%" prompt or a "login" instead of "**", or if the system should crash while you are sending a comment, please contact either of the authors at the Survey Project about your difficulties. These error messages may indicate that text has been lost or not successfully entered in a conference.

Appendix

SAMPLE INTERACTION ON *TC*

As a means of reviewing the main features of *tc*, a sample interaction is presented below in which a new user, Lathrop, begins a session.

```
login: telectr e
Welcome
Name?lathrop e
Conf. eies: 3 unread
Conf. nroff: 2 unread

To see new items type "seecomments <conference-name>"
To enter new item type "newcomment <conference-name>"
To modify existing item type "modify <conference-name> <comment-number>"
To have an overview of all your conferences type "overview"
To see status of conf. participants type "status <conference-name>"
To see this list type "help"
To quit type "--"
```

Lathrop is a member of the conference "nroff". After checking the status of this conference...

```
** status nroff
CONFERENCE: nroff
UP TO DATE              smith
UP TO (NOT INCLUDING) 16 (2 unread) lathrop
UP TO (NOT INCLUDING) 17 (1 unread) mpearson

17 ITEMS
```

Lathrop uses a "seecomments nroff" to see its latest comments; each new comment is displayed after Lathrop presses a carriage return:

```
** seecomments nroff e

Seecomment: Depress carriage return to see next item. Terminate with "--".
e

Conf: nroff             Comment: 16          Lines: 4
Entered by: mpearson   on: Tue May 6 08:57:04 CET 1980
Keys: varian program available

How are things coming on your text editing tutorial?
I have written a short program to simplify sending text
over to the varian.  You can try it the next time you
want to print out your tutorial.

Seecomment: Depress carriage return to see next item. Terminate with "--".
e

Conf: nroff Comment: 17 Lines: 3
Entered by: smith on: Mon May 12 08:26:46 CET 1980
Associated IIASA Comment: 12
Keys: standard macro package for IIASA/text-processing

Let's think about making a macro package that anyone at
the institute can use.  There should be one set of input
conventions for all word-processing here.
**
```

If Lathrop wants a hard copy of any of the comments in the "nroff" conference, s/he can use the "lprintcomment" or "variancomment" commands:

```
**lprint nroff 16
**variancomm nroff 17
**
```

Now checking the status of "nroff" shows that Lathrop has seen everything:

```
** statu nroff e
CONFERENCE: nroff
UP TO DATE              smith
UP TO DATE              lathrop
UP TO (NOT INCLUDING) 17 (1 unread) mpearson

17 ITEMS
```

Next Lathrop decides to respond to one of the latest "nroff" comments by entering a new comment in that conference:

```
**new nroff e
(You are in the IIASA editor.  Type "x" to use edx.
Type "w" then "q" to enter comment.)
ENTERING SCRATCHPAD:
0
*x e
```

Since Lathrop finds it easiest to compose and modify text using "edx", s/he types an "x" in response to the editor prompt, *. This starts "edx"'s full-screen display.

```
_*x_@_________________________________________________________________
|I agree that the institute needs a standard macro package that      |
|anyone here can understand and use. As far as I know, no such       |
|package has been agreed upon yet. I suggest we modify Smith's       |
|package -- it's relatively simple and straightforward.<escape...q>  |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
-                                                                    |
+--------------------------------------------------------------------+
*w @
245
*q @
```

After finishing the comment, Lathrop presses the "ecape" key, followed by the "q" key to leave "edx", then types a "w" and a "q" to begin the sending process. *Tc* responds with a series of questions that will supply information for the header:

```
Associated IIASA Comment (*)? 17 @
Keys (Word/Phrase/)?Standard Macro Package @
Associated Comment: 17
Keys: Standard Macro Package
OK to send (y/n)?
```

Lathrop is satisfied with the comment and decides to enter it in the conference by typing a "y" in response to *tc*'s latest question:

```
OK to send?y @
Comment being entered.
Entered as:
Conf: nroff Comment: 18 Lines: 4
Entered by: lathrop on: Fri May 30 15:37:46 CET 1980
OK to delete SCRATCHPAD? (y/n)?y @
```

As a final step in entering the comment, Lathrop agrees to delete SCRATCHPAD and types a "y" to do so.

Before logging out, Lathrop uses the "terse" command to stop *tc* from printing out a conference overview and a command list at the beginning of future sessions:

```
**terse
**
```

Lathrop knows that the omitted information can be obtained with the *tc* commands

```
**overview
**help
**
```

After this, Lathrop is ready to end the session:

```
**-- e
login:
```

References

Hiltz, S.R., and M. Turoff. (1978) *The Network Nation.* Addison-Wesley.

Kulp, J. (1979) *Computer Services in 1980.* Internal IIASA paper.

Pearson, M.M.L. (1980) *Using the Computer to Communicate: An Introduction to Computerized Conferencing* WP-80-72. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Uhlig, R.P., D.J. Farber, and J.H. Bair (1979) *The Office of the Future* North-Holland Publishing Company.