DIARY OF A HUMAN FACTORS EXPERIMENT

William Seymour

COINS Technical Report
77-14

January 1978

ACKNOWLEDGMENTS

The design of this experiment has been greatly influenced by the advice of others. Discovering and exploiting these sources was, in fact, one of the goals of this project and many specific contributions are mentioned in the diary. At the risk of repetition I wish to acknowledge, and thank for their generous donation of time, the following:

Henry Ledgard, Daryl Winters, and Andrew Singer

Lance Miller, Michael Marcotty, and John Gannon

Chuck Clifford, Dan Anderson, Ram Dahiya, and Conrad Wogrin

I. Introduction

This report will describe a project involving the attempted validation, through controlled experimental testing, of a human factor's principle: *Natural language provides a suitable model for interactive computer language.* The experiment is now in final design but has not yet been performed. The principle goal of this project is not, in fact, the execution of the experiment but more the study of its design. In keeping with proper experimental methodology, therefore, this status report is offered now. In this way the report of the experiment's development is free of any bias resulting from a post facto interpretation of results.

This report is in three sections:

1. A description of the experiment as it will be executed,
2. A diary of the development of the experiment, and
3. An appendix containing the experimental materials.

II. DESCRIPTION OF THE EXPERIMENT

We hypothesize that a text-editing system whose syntax is modeled on natural language will be superior in practical use to one which is not. For this reason, the overriding concern is to create an experimental situation which mimics the real world. We will recruit users to form a sample population representative of those likely to use a text editor. We will randomly assign each subject to one of two experimental treatments: One group will use an editor with syntax determined by natural language principles and the other will use a parallel editing system with more notational syntax. Each subject will then participate in a controlled experiment in which they learn to use the editor and then perform two editing tasks with it. The learning and task performance phases will be timed and the number of syntactic errors committed during task execution will be measured and compared. The experiment is described in greater detail under the headings below:

Experimental design:

Recruiting begins with the collection of demographic information. The person, if qualified, is told that this is an experiment to compare two text editors; the obligations and incentives are explained; and participation is requested. If willing, the person is randomly assigned to one of two experimental sequences. Appointments are made in pairs: A subject using one of the editors will be tested at the same time as another subject from the same subgroup who is using the other editor.

When a subject reports he is given a typing test. He then reads a sheet of general instructions describing the experiment and is seated at a computer terminal. He types "START" to begin the timing and proceeds to study a manual describing the appropriate editing system. As he studies, he may practice using editing requests on the terminal. When the subject feels he has mastered the use of the editor, he enters a command which ends the training phase timing. He is then given instructions pertaining to the task. He types "START" again to begin the task timing and is given a representation of the beginning and ending forms of the file to be edited. The session ends when the text has been successfully edited to match the target. The exact procedures involved can be seen in "Instructions for Task" in the Appendix.

The subject returns by appointment a week later and is given a ten-question, untimed recall quiz. The questions are independent and require the composition of a simple sequence of editing commands to perform a specified function. No reference to the manual is allowed. Following this, the subject performs a second editing task, with manual available, using the same editor and with similar procedures as in the first session. The subject is paid and dismissed. This sequence is shown in Table 1.

TABLE 1: Experimental Design

1) Recruitment and Random Assignment to Sequence 1 or 2

	SEQUENCE 1		***SEQUENCE 2***
2)	Familiarization with OS	2)	Familiarization with RS
3)	Task 1 with OS	3)	Task 1 with RS
	4) One week hiatus period		
5)	Quiz on OS	5)	Quiz on RS
6)	Task 2 with OS	6)	Task 2 with RS
	7) Payment of incentive		

Note: ***OS: "Original Syntax" -- Notational Version***
RS: "Revised Syntax" -- Natural Language Version

The Two Editors:

Both editor versions are based on the UMASS HOPE (Human Oriented Program Editor) system. The notational version of this is modeled after the CDC NOS Version I Text Editor. The available requests represent a subset of the NOS Editor system, with a few small modifications and the addition of two requests normally a part of the operating system. The natural language version contains the same requests but with syntax much closer to the HOPE editor. Error messages, caution checks, and prompting conventions are as in the HOPE system, with necessary modifications. A summary of these editors is shown in Table 2.

The Performance Measures

Each subject has one training period, executes two tasks, and takes a quiz. Two main performance measures will be taken with each task execution:

1) the total time from start of task to successful completion, as measured by the computer;

2) the number of incorrectly formed commands (syntax errors) enroute to completion.

Every session, including the training period, is entirely recorded automatically with the actual time attached to each input and output line. The performance measure for the quiz is simply the number of correct responses (any command sequence which achieves the desired edit is "correct"). These will be hand-scored.

We view the total time for each subject (training period and two task executions) as the principal measurement. The

TABLE 2: Editor Syntaxes

Experimental Versions

CDC NOS VERSION 1 Text Editor	*OS: Notational Syntax*	*RS: Natural Language Syntax*	*HOPE Editor*
FIND(:/text/)(;n|;*)	FIND(:/text/)(;n|;*)	FORWARD (n LINES|ALL LINES) (WITH /text/)	FORWARD (n|ALL)(/text/)
RESET	FIND(:/text/)(;-n|;-*)	BACKWARD (n LINES|ALL LINES) (WITH /text/)	BACKWARD (n|ALL)(/text/)
LIST(:/text/)(;n|;*)	LIST(:/text/)(;n|;*)	LIST (n LINES|ALL LINES) (WITH /text/)	LIST (n|ALL)(/text/)
DLLETE(:/text/)(;n|;*)	DELETE(:/text/)(;n|;*)	DELETE (n LINES|ALL LINES) (WITH /text/)	DELETE (n|ALL)(/text/)
EXTRACT(:/text/)(;n|;*)	EXTRACT(:/text/)(;n|;*)	HOLD (n LINES|ALL LINES) (WITH /text/)	COPY(n|ALL)(/text/) INTO file-name
ADD	ADD	INSERT TEXT AFTER	INSERT AFTER
CHANGE	CHANGE	INSERT TEXT OVER	INSERT OVER
ADD $	ADD $	INSERT BUFFER AFTER	INSERT file-name AFTER
CHANGE $	CHANGE $	INSERT BUFFER OVER	INSERT file-name OVER
RS:/old-text/, /new-text/(;n|;*)	RS:/old-text/, /new-text/(;n|;*)	CHANGE(n|ALL) /old-text/ TO /new-text/	MAKE(n|ALL)/old-text/ new-text/
EDIT, file-name	EDIT, file-name	ASSUME FILE IS file-name	ASSUME FILE IS file-name
REPLACE(,file-name)	REPLACE	PRESERVE	PRESERVE(file-name)
END	END	END EDITING	QUIT

EXPLANATION

1) Keywords are in capitals. They may be abbreviated by their first letter.

2) Variables are: n (a number); text (a text string not containing /); file-name (a file name).

3) | indicates alternatives.

4) Command phrases in parentheses are optional.

5) / is used to delimit all text strings.

total number of syntax errors in the two tasks and the recall quiz constitute two additional and separate measurements. We hope the natural language syntax editor will be superior in all: less time, fewer errors, more correct responses. Further analysis will be done on the relationship of the three component timings that make up the total time for each subject.

The Sample

In an attempt to have an experimental sample representative of the whole population of potential text editor users, the sample is divided into four subgroups. One consists of persons with "secretarial skill" (defined as typing ability in excess of 30 wpm) but without restraint on computer experience. The other three are uncontrolled for typing ability but require specified levels of computer experience:

1) naive (0-5 hours terminal use claimed),

2) familiar (6-50 hours terminal use claimed), and

3) experienced (over 50 hours terminal use claimed).

In addition, we require that all subjects have no prior experience with the CDC NOS Text Editor. The sample is summarized in Table 3.

TABLE 3: The Sample

Factor A: Subject Experience	Factor B: Editor Version		
	Level 1: Natural Language Syntax	Level 2: Notational Syntax	Total
Level 1: computer-naive	7	7	14
Level 2: computer-familiar	7	7	14
Level 3: computer-experienced	7	7	14
Level 4: clerical-experienced	7	7	14
TOTAL	28	28	56

Statistical Considerations

From a statistical standpoint, the experiment to be conducted will be viewed as three separate experiments. All three will be of randomized, fixed factor design with factors and treatment levels as previously shown in Table 3. While we will be concerned with interaction effects and the main effects of Factor A (Subject Experience), our primary concern is with the main effects of Factor B (Editor Version). We will use standard analysis of variance techniques in an attempt to reject the null-hypothesis that the two versions, as tested by the three performance measures we have chosen, are equivalent. We will use a two-tailed test for this reason: While we hope the natural language version will be superior, the more notational version may have advantages for users accustomed to this style. The only statistical consideration tying the experiments together is our supposition that the three performance measures should correlate.

It was noted earlier that some additional measurements (more detailed task timing) will be recorded. These will not enter into any statistical analysis but will be used to suggest possible directions for further experiments. The interest in future research also underlies the decision to use the four levels of subject experience despite the small cell sizes thereby implied.

III. DIARY: DEVELOPMENT OF THE EXPERIMENT

***06/07/77 Tuesday

I am working this summer as a research assistant for Henry Ledgard. The research, sponsored by a National Science Foundation grant, involves human factors in Computer Science. As one of my first duties, I have been reviewing literature in this field and I came across an article that commented critically on the quantity and quality of actual experimentation. My literature search supports this contention: Very rarely are proposals for improved human factors design validated by psychological testing and when the attempt is made, it is usually characterized by sloppy methodology.

***06/10/77 Friday

Henry is very aware of the lack of human factors experimentation and has been encouraged by Bob Graham, the Department Chairman, to consider projects in this field. Nico Spinelli, with the strongest behavioral science background of the computer science faculty, is another proponent of this type of work. Henry suggests that I do such an experiment as my Master's Project. He is particularly interested in validating the principle that natural language is the best model for interactive computer languages. We agree this would be the basis of a reasonable experiment and determine to spend a little time each day "brainstorming" the matter.

***06/14/77 Tuesday

We can proceed in either of two directions with an experiment: analyze a large amount of already existing data for gross trends or collect a smaller amount of data, tailored to our needs, and analyze it in greater detail. The languages we can look at are the operating systems language (NOS), the text-editors, and the programming languages.

***06/17/77 Friday

One method of collecting smaller amounts of data for careful analysis is to collect listings of actual user sessions and hand check these. One question we could examine would be: Are more errors made using requests that violate natural language syntax? The listings could be collected by scouring the terminal rooms for discarded listings, with rules to assure an appropriately random sample. The most fertile area seems to be the use of text editors: users range from naive to professional and the syntax of the commands involved offers some hope of analysis. After some discussion, Henry and I conclude that adequate control of this type of experiment would be an insurmountable problem. The search goes on.

***06/20/77 Monday

An investigation of saved system messages also fails to yield any possible experimental data. The only information routinely saved by the system is the log of dayfile messages. Only syntactically correct commands are saved, making error analysis impossible; and the NOS language itself isn't sufficiently interesting from our point of view. No record is kept of text-editing sessions or compilations, and the possibility of installing an internal monitor to trap commands is considered too disruptive by the systems people.

***06/21/77 Tuesday

Repeated dead ends of the sort indicated above have led us to the conclusion that a small-scale, controlled experiment is the way to go. With such a free hand to design an experiment to generate data to our specifications, we waste a good deal of time before we recognize the realities of the situation. What we would ideally like must be constrained by limited resources of time, money, and

manpower. We would like a large sample but realize the need for incentive payments must limit us to about one hundred subjects. We would like to observe computer use over time but realize we must make the experiment a "one-shot" proposition.

***06/23/77 Thursday

We have decided to test the human factors principle: "Natural language provides a suitable model for interactive computer languages." We now must decide how. In our earlier search through discarded terminal listings, we had focused on text editors. This dovetails with Henry's work on the HOPE (Human Oriented Program Editor) system which is in the final implementation stages. Using the semantics of this system as a basis, we could modify the syntax to our purposes and record subjects experience with it. Two experiments are being considered: 1) Design one editor with a mix of "good" and "bad" commands and compare error frequency within subjects. 2) Design a "good" editor and a "bad" editor and compare learning and use results across subjects.

***06/27/77 Monday

We now focus on the natural language principle itself: What is "good" syntax? The HOPE system as designed embodies the principles we want to prove correct so seems a reasonable basis for our "good" commands. Generating "bad" commands must be done in a manner to preclude accusations of bias so we decide to model them on an existing and widely used text editor. We choose the NOS Text Editor. Carrying this idea forward we work on an editor version that has a mix of HOPE and NOS requests and soon hit a snag. If we provide duplicate requests (having different syntax but the same function) the point of the experiment is immediately obvious. If we have only one request for each function, with some functions implemented with "good" syntax and some with "bad," then the error rates observed may be caused by the difficulty of the function rather than the effect of the syntax. For this reason we reject the "mixed" experiment.

***07/01/77 Friday

We have decided to go with two versions of the editor and an "across-subject" experiment. Henry and I, perhaps belatedly, lay out the following guidelines:

1) The editor must be simple enough so that it can be learned fairly quickly.
2) The editor must be sufficiently powerful to perform a reasonable editing task and not be just a "toy."
3) The semantics of the editor's command language must be sufficiently complex to allow the natural language modeled version to demonstrate its presumed superiority over the non-natural language modeled version.
4) The non-natural language version should not be set up as an obvious "strawman," i.e., no contrived unnaturalness should be installed to insure its poor performance.
5) Both versions should be internally consistent and logically complete.
6) There should be a one-to-one correspondence of functions in the two versions: If a command sequence in one version performs a certain function, there should be a command sequence in the other version to perform the same function.
7) The two versions should require an approximately equal number of key strokes to execute identical functions.

Our first attempt at the two versions, one like the original HOPE system and one a subset of NOS commands, is not perfect but encouraging.

***07/11/77 Monday
I talk with Daryl Winters, who is implementing the HOPE editor. We must be careful to design the two syntaxes so that they can be implemented with only minor modifications to the basic HOPE editor. Daryl describes his parsing routine which seems to be more than sufficiently flexible to accommodate the changes we are considering.

***07/12/77 Tuesday
With the first flush of success over hitting on a viable experiment comes the sobering realization of how many issues are unresolved. The next problem to tackle seems to be the exact nature of the experiment: What do our subjects do with the two versions of the editor? We want the experiment to be as "natural" as possible. Then, if the results bear out our beliefs, we may conclude that the natural language version is superior for real editing by real users. This suggests the performance of an editing task rather than the administration of a questionnaire: We want to find out how people react to the editors, not how they perform on quizzes. The rough plan is, therefore, to train subjects to use the editor by allowing them to study a manual and then to record their performance on a task. Half of the subjects will use one editor version and half the other. Thus committed to a controlled psychological experiment, it seems reasonable to seek advice from a trained psychologist.

***07/15/77 Friday
I talk today with Chuck Clifford of the Psychology Department. The conversation is a general one about the process of designing a psychological experiment and hits on a number of useful points that will often recur in the design of this experiment. At this stage his most valuable input is the notion that an effort to be consciously aware of all design considerations will help remove any bias from the experiment.

***07/18/77 Monday
What Chuck Clifford said recalled my dissatisfaction with the methodology used in the computer-oriented human factors experiments I have reviewed. I bring this up with Henry and we enthusiastically agree that the process of designing a psychological experiment, rather than the experiment itself, would make an excellent project. We will actually perform a meaningful experiment but the emphasis for my Master's project will be on a diary recounting the decisions that went into designing the experiment. We hope this will prove valuable to other researchers doing similar work.

***07/21/77 Thursday
Henry and I discuss a little general theory. We want to compare the effect of two controlled experimental treatments: The natural language syntax versus the more notational syntax. Since we will be measuring peoples' performances to look for these effects, we can expect "noise" to be introduced by the different levels of human skill and motivation. These factors must either be explicitly controlled or effectively randomized if we are to have an unbiased result. If these factors are randomized, we must assure that the variability thus introduced is less than the variability caused by the experimental treatments. We could lose any evidence of the effect we seek if any of the following were true:

1) The different syntaxes are too small or insufficiently complex to cause any detectable difference in performance.
2) The performance being measured is too short to yield measurable

differences.

3) The sample is too small.

***07/22/77 Friday

I am particularly concerned that the effects of the two different syntaxes may be masked by the different skill levels of the subjects. Digging back in my weak and well atrophied statistical background, I devise elaborate schemes to measure intelligence for use as a covariant. Reason returns and I call the Mathematics Department for help.

***07/27/77 Wednesday

I talked earlier with Ram Dahiya of the UMASS Mathematics/Statistics Department. He listened politely to my covariant designs and carefully shot holes in them. However, he was very optimistic about the chance of finding statistical significance (if the effect is actually there, of course) in a straight-forward randomized design. Not completely convinced this was adequate, I next seized the idea of having each subject use first one, then the other editor. I would then combine the total results, factor this to the group mean for total time, and compare differences for the editors within each normalized subject. I raised this possibility with Dan Anderson of the Psychology Department, who pointed out the confounding influence of "learning transfer effects" and the like. He, too, argued for an uncomplicated randomized design, and I now capitulate. With this settled, the statistical work will be relatively simple. One final issue is whether to use a one- or two-tailed test. Anderson and Dahiya agreed that there wasn't sufficient a priori evidence to require a one-tailed test so we will adopt the less conservative two-tailed test.

***07/29/77 Friday

I turn my attention next to determining the experimental sample. We wish to test the hypothesis that our natural language syntax editor is superior but that raises the question: "Superior for whom?" Our sample, we decide, should include a cross-section of people who would be likely to use a text editor. This would include secretaries, students, professional programmers, and researchers. If we recruit randomly, we have no guarantee of getting a suitable mix of user types, and if we recruit by category, our total sample restriction will impose small cell sizes. In the latter case, can we legitimately aggregate results from each subsample to reach an overall conclusion? How small can the cells be and still remain statistically meaningful? Time for another visit to the Statistics Department.

***08/02/77 Tuesday

Ram Dahiya describes a statistical procedure by which results can be normalized for each subgroup, thus allowing aggregation. It is apparently a routine procedure and thus in keeping with my new resolve to hold statistical complexity to a minimum. I am surprised to learn from him that subgroup cells as small as five subjects will yield analyzable results in many experiments. I double-check this with Dan Anderson, who concurs. I talk with Henry about this and we tentatively agree on four user types: three different levels of computer experience plus one group of subjects with proven secretarial skill. We think this will assure a sample roughly representative of the population of potential editor users. We envision equal cells but haven't yet fixed a total number. That will be largely determined by how much incentive money we can afford, and we don't know yet what we will have to pay each subject for cooperation.

***08/04/77 Thursday

We will be testing some subjects who have never used a text editor before and who have no computer experience. I am concerned that the presentation of the editor in the manual, no matter how well written, may be too much for some subjects. For this reason, I would like to be allowed to answer questions during the manual familiarization period. This, of course, raises all kinds of biasing problems and will require some thought and advice.

A similar question is what happens if a subject accidently deletes the whole file or makes such disastrous errors that he becomes frustrated and quits? The choices then are: 1) assign some maximum time, 2) replace the subject, or 3) throw out this subject. The first solution seems much too arbitrary and the second, considering the recruitment/assignment procedure, would seriously contradict our randomization methods so it appears that we are left with throwing out the subject and having a reduced sample. This could be a real problem in view of our already small cell sizes so all steps to avoid this circumstance must be taken.

***08/09/77 Tuesday

Thus far we have avoided the issue of how to score the results. We want to show our editor is "better" in normal use so envision having the subjects first train, then perform an editing task. Henry, Andy Singer, and I discuss this plan and agree that the most meaningful measurement is the length of time to do the task. Andy is in favor of keeping track of other details, such as "think" time for the different requests, the ratio of syntactically incorrect requests to correct requests for each type, etc. I work against the tendency to try to test everything since this would overly complicate the experiment. We compromise: The measurements for statistical purposes will include only the time taken to complete the task and the gross number of errors. If additional data can be saved without disruption, it will be done but used only to interpret results and plan possible future experiments.

***08/10/77 Wednesday

Henry, Andy, and I continue to debate the measurement issue. "Normal" use of an editor would presumably be characterized by sporadic use. Thus a "good" editor would have a syntax easily remembered after periods of disuse and we feel our natural language model will have a real advantage here. This, of course, would not show up in our simple timing. John Gannon had raised the possibility of recalling subjects after a month for a retest. We do not want to stretch out the experiment that long so we agree on a one week hiatus and consider what form the retest should be. We would like to have the subjects do another task without permitting reference to the manual. The abort rate for this test, however, would be very high so we decide to give a "closed manual" quiz. This will consist of descriptions of editing situations: The subjects must form the requests to do the necessary editing. Since the subjects have to come back anyway, we decide we might as well train them with the other editor and have them do a second task. The scores on the second task clearly cannot be combined with the first so will stand as a separate and secondary experiment that will provide results admittedly hard to interpret.

***08/11/77 Thursday

The arguments about what information to collect point up one of the biggest lessons I have learned in my conversations with the psychology people: All a priori assumptions and all statistical

plans must be stated before the experiment is run. The temptation to explain an unexpected result with "free" data is hard to overcome. Thus, although I yield to Andy's requests to collect any data that may be useful in directing future experiments, I confirm my resolve to apply preplanned statistical tests to only the data designed for these tests. I have been studying EXPERIMENTAL DESIGN by Cochrane and Cox and feel comfortable with the statistical analysis.

***08/15/77 Monday

Today I talk to Lance Miller and one of his first concerns is the nature of the tasks: Will they be program segments, formatted text, or unstructured character strings. The editor on which we are basing our two versions was designed to edit PASCAL programs but can easily be used in other applications so the decision will be based on the desire to recreate a "natural" use of the editing system. In view of the diverse subjects we will be using, we decide to use text. A book of one page anecdotes by Bennett Cerf proves to be a ready source. These will provide a little interest without being too distracting, a fear that had argued against giving programs to programmers (who might disregard the task and study the code). This still leaves the question of how long to make the text and how many errors to put in. We want the task to take long enough to reflect editor differences but a high density of necessary corrections will be confusing and a long text will penalize those who list the whole file. We realize we will have to wait till the pretest phase to "tune" the tasks to the subject population.

***08/19/77 Thursday

I talk to Chuck Clifton of the Psychology Department again. One issue we discuss concerns the testing environment: Should the test take place in a "natural" environment, for example in a terminal room where there are other users, or in a "laboratory" environment such as a terminal in a private office would provide. The issue here is a choice between randomization and control. We conclude that using a public room will present too many scheduling problems and opt for the sterile, controlled situation.

***08/23/77 Tuesday

The experiment is shaping up well but before going to the next level of refinement, Henry and I have lunch with Dan Anderson to discuss the whole project. The chief topic of conversation turns out to be the two syntaxes themselves, and Dan has an interesting question: On what grounds do we claim that our natural language version reflects natural language? Since it is true that we did not follow any stated principles in designing the syntax, we will not be able to generalize the result. The criticism is valid but since any attempt to do that now would clearly be post hoc fudging, we can only proceed.

***08/24/77 Wednesday

We sketch a rough timetable for the project. While I await implementation of HOPE, I will finalize the two syntaxes and prepare the materials. Assuming HOPE is up by October, we can do the implementation of our two editor versions and begin pretesting in November. We then can do the actual experiment in December.

***08/26/77 Friday

I return to work on the two versions, enlisting Daryl's help to avoid any implementation snags. It soon becomes apparent that the earlier work wasn't thorough enough and that there are conflicting

goals to resolve. A round of phone calls leads to a consensus that a strict adherence to parallelism and internal consistency is more important than the faithfulness of the NOS version to its model. As long as any changes only improve the sense and power of the NOS version, it will not violate any of the guidelines set earlier. Another conflict concerns the need to choose keywords for the "good" editor that will be mnemonically neutral, since we want to test only syntax differences. This is complicated by our desire to retain the first letter abbreviation rule. On this subject we are conservatively modifying an earlier guideline: The two versions only require an equal number of keystrokes if the abbreviation rule is used fully. The NOS version will take fewer keystrokes if subjects don't abbreviate and thus we are safe from claims of unfair biasing.

***08/29/77 Monday

An important detail is considered today: How honest to be with subjects about the purpose of the experiment. The psychologists advise that the best policy is to be as open as possible within the constraint that full knowledge of experimental purpose could lead opinionated subjects to alter their performance to match their beliefs. We decide to merely state that we are "evaluating two editors." A similar issue concerns the names of the two editor versions. Since materials will be labeled the names must not indicate our bias. We have been calling them HOPE and NOPE and now choose the neutral initials RS (for revised syntax--the natural language version) and OS (original syntax--the NOS version).

***08/30/77 Tuesday

We are now able to settle a question about sample. We have been considering the inclusion of one subgroup of NOS users while requiring all others to be naive with respect to that editor. Now that OS is no longer a pure subset of the NOS system, it isn't apparent whether previous NOS use would be a help or hinderance to an OS subject. Therefore, we will require all subjects to have no prior NOS experience. We realize this may make recruitment of ex-perienced computer users difficult.

***09/07/77 Wednesday

Not much thought has yet been given to the materials which will be required. The most obvious, of course, are the manuals and the tasks. In addition, the recall quiz will have to be prepared and instruction sheets provided.

The manuals are critical: They have to be relatively concise and well-written but must also lend themselves to a perfectly parallel presentation for both versions. I begin work on the RS manual first, using the HOPE manual as a model. The goal is to write it such that the OS manual can be generated by merely replacing any reference to an RS request by the parallel OS syntax. In doing so, I have to be careful to use neutral verbs in describing functions so as not to bias towards a keyword in either language. Perfect parallelism is not possible (due to the difference in ellipsis treatment, for example) but I am satisfied that, where differences occur, there is no bias. Writing and polishing the manuals proves to be very time-consuming.

***09/09/77 Friday

The manuals are in two sections. The second describes the function and use of each request and will be different for the two versions. The first section, identical in both versions, is an attempt to acquaint a computer-naive subject to the use of the

terminal and the concepts involved in text-editing. It proves to be difficult to write.

***09/14/77 Wednesday

As a break from working on the manuals, I develop the other materials. The first is a page of general instructions which introduces each subject to the ground rules and his commitments. In the test session, I will distribute this first then give the appropriate manual to the subject. When the training phase ends, I will hand over detailed instructions for the task, and then finally the text itself and editing requirements. I write the task instructions knowing that they will not be final until Daryl completes the implementation. Henry insists that all materials be coded as text files. If no other purpose is served, I at least get lots of hands-on experience with text editors.

***09/16/77 Friday

An example of how details can kill you occurs today. In designing the materials for the task itself, I decide to present three pages of text. One will show the beginning form of the file, one the target form, and one will indicate the required editing. This last page will be a copy of the beginning form with penned in editing marks. A seemingly easy job, but it turns out that the required editing page must be double spaced for legibility. This requires a caution to that effect in the task instructions. Also, there is a blank line between paragraphs in the beginning text which could be confusing. Finally, I think I get them right.

***09/19/77 Tuesday

A little more thinking about the sample. We earlier decided on four groups: one group with secretarial skill and three with different levels of computer experience. We decide that a typing test will be the arbiter of "secretarial skill" and that "computer experience" will be a function of hours of terminal use claimed. We now decide on 14 subjects per group (or seven per treatment within group) making a total of 56. These figures have no special statistical significance and represent the best compromise between what we feel is the minimum sample for statistical validity and the maximum we can afford in time and money. As our plans become more elaborate, we must often remind ourselves that the point of this experiment is more to illustrate experimental design than to produce an earth-shaking result.

***09/27/77 Tuesday

I have a few friends read the manuals for critical comment. Among the changes suggested the best is to add an annotated sample editing session. I'm embarrassed I didn't think of that before and go to work on one for each editor version. Each will start with the same text and use parallel requests. By including intentional errors I can prepare the subjects for error messages they will see.

***10/12/77 Wednesday

The HOPE editor is now almost ready. Daryl gives me access to the latest version so I can begin using it. My first concern is to see what error messages and caution checks it provides. We want OS and RS to yield the same responses and to list text in identical fashion. We plan to use whatever is implemented in the basic HOPE version unless this causes a problem.

In HOPE there will eventually be a user option to select TERSE or VERBOSE mode. The current version uses VERBOSE mode which is fine for our purposes, since the subjects will be new to the editor and presumably would prefer full messages.

***10/14/77 Friday

A small sample problem pops up: We have a subgroup labelled "skilled secretarial" for which we will screen to assure a minimum level of typing skill. Should we require no typing skill in other groups and screen it out? This would be a problem since experienced computer users (by our definition: "over 50 hours of terminal use") may test well in typing. We decide to test everyone but only use it as a screening device in the one subgroup. More "free" data for Andy Singer.

***10/19/77 Wednesday

Daryl reports that the basic HOPE editor is now fully implemented and 98 percent debugged. He is thus ready to work on creating the two modified versions for my purposes. I give him the one page summaries of each syntax and we discuss the other programming requirements that are implied by the task set up. I envision the creation of a procedure that I will call, supplying subject number and editor version. The subject will then be seated at the terminal, will type "START," and the timing will begin. The subject must call the file, do the necessary editing, save the file, and end the session. When the file is saved, a checking routine will be activated which will inform the subject that the file matches the goal text or that it does not, display the first nonmatching line, and ask the subject to continue editing.

***10/21/77 Friday

A problem that has been a nagging worry yields an acceptable solution today. System response time is an unwanted independent variable that is difficult to control. One solution, that of subtracting out system response time from total task time, not only fails to solve the whole problem--user response time is undoubtedly a function of system speed--but also is virtually impossible to implement on the NOS system. If we are able to pair sessions, however, so that an OS subject and an RS subject from the same subgroup are tested at about the same time, then this will assure unbiased results. This implies that we must recruit subjects, determine their subgroup, randomly assign them to an editor version, and then recontact them to make an appointment. Complicated but apparently necessary.

***10/24/77 Monday

Daryl has the RS version ready for trial. There are only a few bugs and I'm amazed at how quickly it is ready. I recruit a COINS graduate student to try one of the tasks using RS and the results are encouraging. Daryl wants to work on the programming to implement the task set-up before doing the OS version so we discuss the best way to capture the information we need: As a bare minimum we want the total elapsed time for the task and a count of improperly formed requests. Although they will not be part of the statistical

conclusions drawn, counts for each type of request and timings indicating "thought" time are also desirable. The obvious solution is to record all input (user requests) and output (editor responses, including error messages) with the actual time attached to each. This will generate a lot of data so an indirect access file is indicated. The possibility of generating one file for each subject is considered and rejected in favor of one large file with a fixed record format. Each record will contain this information:

1) subject number,
2) session number,
3) date and time,
4) code to indicate whether input, error message, or text output, and
5) text.

***10/26/77 Wednesday

Daryl and I continue to talk about the program to implement the task. We are a little concerned that the timings recorded in the saved files are not exactly what we want! Some system processing is done between the time the subject presses the "return" key and the time when we can interrogate the clock. Although this delay will be dependent on system load, it turns out that not only will it never be too large but it is also consistent for input and output. Thus, interval timings from prompt to response will always be accurate.

While on the subject of computer speed, I ask Daryl if either the OS or RS parser has a built in speed advantage. He assures me the implementations are perfectly parallel and parsing speed is only a function of the length of the input line. Since this has been attended to, all is in order.

***10/31/77 Monday

Andy Singer and I talk about how I will eventually recruit the subjects. I expect to use all of these methods:

1) Advertise in the UMASS newspaper.
2) Post notices around the University Computing Center.
3) Word of mouth and personal recruiting among secretaries in the Graduate Research building.
4) "Voluntary" recruiting of Introductory FORTRAN students by the teaching assistants.

Since we will be taking subjects at their word about computer experience and nonuse of the NOS text editor, questions about these qualifications must be presented in a manner to encourage truthfulness. The incentive payment should be high enough to encourage subjects to return for the second session but not so high as to attract "paid workers."

11/07/77 Monday

Daryl discovered that we had overlooked conflicts for the first letter abbreviation rule in the OS version. This is due to the fact that we are including operating system commands (for example, REPLACE) within the text editing system itself. We can resolve this by requiring a full spelling of NOS keywords where that was required in the original. This corruption of the one-letter abbreviation rule penalizes OS but, at least, we can argue that our version is faithful to the original. With these changes, Daryl can finish the OS version. I now have two parallel procedure files, one each for OS and RS, that accept subject and session number as input, administer the task using the appropriate editor, and append data from the current session to an indirect file. We are almost ready to pretest.

***11/15/77 Tuesday

It has proven to be unusually difficult to find a "professional" typing test. I finally arrange today to get one from Mrs. Frederickson, a typing teacher in the Amherst public school system. I plan to switch the terminal to LOCAL mode and test each subject at the beginning of the first session. Based on the norms supplied with the test material, Henry and I decide that 30 words-per-minute will be the qualifying cut-off for subjects recruited for the "secretarial skill" subgroup.

***11/18/77 Friday

As the pretest phase approaches, I realize how many small details have yet to be resolved and I practice the sequence of administering the typing test, handing out materials, etc. I arrange the set-up of the testing room so that the subject will have a comfortable location and dig up a stop watch to time the typing test and manual reading period.

***11/21/77 Monday

I talk today with Dan Anderson about the possible bias introduced by answering questions during the manual studying phase. We consider and reject as impractical the possibility of hiring someone to administer the test who is unaware of its purpose. The effort of tape recording all sessions also doesn't seem justified. I resolve to wait and let my experience during the pretest determine whether or not questions are allowed.

***11/22/77 Tuesday

I add a useful modification to the testing procedure today: I will have the terminal active with a file logged in while the subjects are reading the manual. In this way, they can experiment with the requests as they read about them. To create more familiarity, I will use as the file the same text used in the annotated session. This will cause a few small hitches in calling the testing program but if it seems useful, Daryl can modify the program to accommodate this change.

***11/23/77 Wednesday

As I practice using the OS and RS editors, I become disenchanted with two of the responses inherited from HOPE. One is a caution check when saving files that may confuse our test subjects; the other is a basic inconsistency in the way the current line is reported that I feel could confuse any user. Daryl agrees to make the necessary modifications.

***11/28/77 Monday

With pretesting almost at hand I call Dan Anderson for advice. He underscores the spirit of the pretesting phase: It is to test the test itself. One of my concerns is whether I can shortcut a few formalities since the first subjects will not be complete strangers to me. Dan suggests that formality is needed only to test the formality itself and reassures that since these results "don't count," I can afford to experiment with different approaches with the first subjects.

***12/2/77 Friday

I attempt to recruit for the pretest. After some experimentation I hit on the strategy of announcing in the terminal rooms that I am looking for anyone who has not used the NOS test editor. Due to end-of-semester frenzy and widespread familiarity with NOS ten

tries net only one subject. The "Experimentor's Form" worked all right as a screening document but I realize I should have included a page to be handed out as an appointment reminder.

***12/5/77 Monday

Henry, Andy Singer, and I spend a day going over all the materials. Their criticisms are valid and counter the tendency I have of over-explaining points where I fear confusion. We agree, however, that I will pretest with the materials as they are and then incorporate the revisions the pre-test dictates with these.

***12/06/77 Tuesday

We have been pushing forward with the project in the hope of running the actual experiment before the end of the year. This now appears impractical so I call a meeting and we agree to postpone the testing till the start of the new semester. At this time, student subjects will be under less academic pressure and thus easier to recruit and the computer system will be less burdened and more responsive.

12/07/77 Wednesday

I did the first official pretest today with Gwyn Mitchell, a COINS secretary. If I had any doubts about the value of pretesting before I certainly don't now. One major error showed up: The file I called to be worked on didn't match the file shown in the "Beginning Text" handout. As a result, things were sufficiently confused that the session had to be ended before task completion. I still learned a lot. Encouragingly, the basic procedures seemed sound and the level of difficulty of the task appeared to be reasonable. A number of small problems reflected my lack of attention to detail and/or inexperience:

1) With the terminal in LOCAL mode for the typing test the carriage return doesn't advance the paper.
2) With the room set-up as it is I must disturb the subject to leave the room.
3) There is an uncomfortable period while the subject waits as I initiate the task program.
4) I have no appointment slips for the second session.

After the session, I asked Gwyn to comment freely on the experiment. She pointed out a couple of weak spots in the manual but claimed that generally she was able to understand the editing system and the task instructions. Before her session was aborted by our discovery of the wrong starting text, she had appeared to be fluent with the editing requests. We scheduled the second session for a week later.

***12/08/77 Thursday

Last night I repaired the starting text in preparation for the second official pretest. Today's subject was a FORTRAN student recruited from a UCC terminal room. He studied the manual and practiced requests for over an hour and had worked on the task for almost an hour when he had to leave for a class. Considering that everything else seemed to go smoothly, I was very surprised to have the test take that long. While I suspect that this was an unusually slow subject, it still appears that the task used today is too long. More serious is the great discrepancy in training time! Gwyn studied the manual 18 minutes and did not experiment with any requests. Although Gwyn appeared to have learned better despite this, it seems that allowing unlimited training time may so increase the noise caused by different motivation levels that the experimental effect

will be masked.

***12/12/77 Monday

Pretesting continues. No more major problems surface and I smooth out the procedures and minor details: setting the subject installed at the terminal, the distribution of materials, etc. The original task is definitely too long so I'm using a shorter one. Daryl's system to save information from the sessions works fine and things are generally encouraging. Andy Singer wants to look at the results and I have to remind him that results are not the point of the pretest.

***12/13/77 Tuesday

One consistent confusion concerns the way the printing head of the Decwriter terminal moves aside when not printing. It takes some getting used to and seems to penalize inexperienced users unfairly. I conclude, however, that nothing can be done to improve the situation. We can only hope that the confusion randomizes.

Another observation is that abbreviation is rather unpredictably used in the RS tests. Since typing out all the keywords penalizes RS users in terms of required keystrokes, I ask subjects about this. They confess to a little uncertainty about the universal application of the one-letter abbreviation rule, but also claim that complete spelling is somehow re-enforcing. This is, of course, the sort of effect we are looking for but I still feel it necessary to alter the manuals a bit to underline the abbreviation rule.

***12/14/77 Wednesday

Gwyn returns after the week's hiatus to do the first second session. Since it is so much like the first sessions, all goes well and she completes the task in a record time of 16 minutes. I still don't feel the task length is resolved: I'm afraid this last one may be too short to show differences. On the other hand, longer tasks are discouraging to the subject and, if the incentive isn't high enough, they may not return for the second session. Long tasks also imply a second problem: It would be necessary to allow two hours for each session and this will mean only four subjects a day during the actual testing period. If this is the case, and if I do the whole thing myself, the experiment will extend over a six-week period. This is undesirable from the standpoint of maintaining constant motivation (which I expect to fall as students start term work) and system speed (which will also decline as the semester progresses). Two alternatives are to hire another experimentor or to drop the second task.

***12/15/77 Thursday

The FORTRAN student does the second session today. As I had suspected, he is unusually slow but he finishes without hitch. Again, however, he took over an hour to study the manual. I am very concerned about this and am considering possible modifications. If we fix a maximum time then there will be less noise introduced by motivation and more by intelligence: which is preferable? Another approach would be to distribute all materials at the outset and then measure total time from start of learning to completion of task. Since we feel that the natural language version should be easier to learn as well as use, this change would lead to a valid but quite different experiment. The difficulty is that this requires the user to plan a strategy: When should he stop studying and start the task? This question is not only important but also involves issues

with which I don't feel comfortable. I resolve to seek advice.

***12/19/77 Monday

I modify the manuals and other materials based on Henry and Andy's earlier comments and on what I have learned in the pre-testing. My attempts to get advice on the motivation problem are frustrating since everyone is gone till January. I decide to stop pre-testing and work on the diary write-up until I can get some advice. After the decisons are made, I then plan to do another round of pretests. This will also be the final chance to "tune" the length and complexity of the tasks.

***12/21/77 Wednesday

Four important issues remain unresolved: the length and complexity of the task; whether to allow unlimited training time; what to do about the second session; and whether to answer questions during training. The first will not be resolved until the final pre-testing round and I await advice on the next two. My limited pre-testing experience has thus far not helped solve the question problem. Subjects availed themselves of the opportunity to ask me questions very infrequently. Questions asked were of a simple nature: I'm afraid subjects may be embarrassed to display ignorance by questioning me on basic problems. If this continues in the final round of pretesting then I will eliminate question answering from the actual experiment.

***12/28/77 Wednesday

I talk with Henry today by phone. We discuss the training time and second session issues and agree to await input from Michael Marcotty, Lance Miller, Andy Singer, and any psychologists I can buttonhole. These are important decisions and will seriously affect the credibility of the project. Furthermore, in view of our plans to distribute for comment the final version of the proposed experiment before executing it, time is getting close.

***01/04/78 Wednesday

I continue to work on resolving the problem issues. I talk with Michael Marcotty and he strongly urges me to talk to a psychologist. He is reluctant to offer advice but leans in favor of unlimited, but measured, training time. He points out possible interesting relationships between training time and task time: perhaps the more accessible natural language version would fool learners into thinking they were proficient before they really were.

***01/05/78 Thursday

I talk today with Chuck Clifton of the UMASS Psychology Department. He recommends keeping the first part of the experiment basically as is but with a separate formal timing of the training period. In conjunction with this, the instructions would be modified to point this out and admonish the subjects to: "learn the material as quickly as possible but be sure you have a complete understanding before beginning the task." He feels this offers these advantages:

1) This most closely mimics the real world. Here, presumably, speed of learning, speed of use, and strategy (when to stop learning and begin use) will all come into play. The experiment will reflect this.
2) This sidesteps the theoretical issue of whether motivation or intelligence randomizes better. The same mix as in the real world will be measured.
3) With separate formal timings no data is lost. Since this

experiment should really be considered as a pilot project in this area, there will be more information to suggest further work.

4) This setup puts a mimimum of artificial controls on the experiment.

With regard to the second half of the experiment, Chuck agrees with me that the cross-training retest is of marginal value. He thinks the closed manual recall quiz would be valuable and, if time permits, he recommends having subjects perform a second task using the same editor. This should be done without a training period but with access to the manual.

I discuss with Daryl the implementation of a task procedure in keeping with these changes. We reach what seem to be reasonable instructions and procedures that present no implementation problems.

I intend to continue seeking advice on these issues but admit that Chuck's recommendations agree with my thinking and are very well supported.

***01/06/78 Friday

I talked to Lance Miller briefly this morning. We discussed the training time issue and he took a new approach. He recommended administering a "programmed learning" type training program that would lead the subject through a controlled practice session. In this way, each subject would be assured of a base level of familiarity before starting the task. The time required, variable since subjects would be required to "do it till it was right," would be recorded and used as a dependent variable. We agreed to talk again next week.

Later I talk to Dan Anderson. His views, appropriately, coincide with Chuck Clifton, the other psychologist. Dan is particularly adamant in doing away with the cross-treatment retest. He feels the positive learning transfer caused by the previous experience with the terminal, the task procedure, and editing concepts will interfere with the negative learning transfer of the new syntax and cause uninterpretable results.

I begin to feel strongly committed to the psychologists' position and prepare my arguments to present to Henry.

***01/10/78 Tuesday

I present my case to Henry today. He completes the near consensus and we agree to proceed as follows: The first session of the experiment will remain unchanged except for formally timing the training period. The instructions will be modified slightly to reflect this and to force subjects to make the strategy decision of when to end training and begin the task. The second session will be altered, deleting the second training session and having the second task done using the same editor as in the first task. Reference to the manual will be permitted during the second task but not allowed for the recall quiz which will precede it. In order to keep the total time span of the experiment within reasonable limits I will hire someone to help administer the tests. The last decison brings with it this bonus: We can now schedule OS and RS sessions at identical times so that system load variations will have no effect.

***01/11/78 Wednesday

I revise the instructions today and decide to deny subjects the opportunity of asking questions. If a problem appears in the final pretesting round I will reconsider but my desire for experimental purity is currently overweighing my fears that there may

be mass confusion.

Another serendipitous result of the new procedure is the resolution of the task length issue. Since subjects will be using the same editor twice there is no need to have same-length tasks in both sessions. Thus we can balance the length of the two sessions and have a wider range of experimental situations if we give a relatively short task in the first session and a longer one in the second. Presumably this will also encourage a higher return rate for the second session.

The statistical considerations have now changed, of course. The main measurement will be the combined total time for the training session and the two tasks. Additionally, each component will be a separate and secondary test. I expect there will also be some interesting relationships between training and task times (compared within subjects), but will make no formal a priori hypotheses regarding these.

***01/13/78 Friday

Daryl and I work out the new task administration program. In order to move towards Miller's "programmed learning" situation I have rearranged the presentation of requests in the manual so that practicing commands in the order in which they are described will result in a mock editing session. The implementation is as follows: Subjects read the general instructions then sit at the terminal. Automatic timing is started by typing "START" at which point they are given the appropriate manual. The first request described is the one to name the file. Any attempt to practice a different request before using this one successfully generates an error message. If the subject uses the request to end a session he is asked whether he is ready to begin the actual task. If he says "no" the end request is ignored and he may continue practicing. When he says "yes" the timer is stopped for the training session and task instructions are distribued. Typing "START" again begins the timing of the task and signals me to distribute copies of the text to be edited. The timer records the time when the task is finished.

***01/16/78 Monday

Daryl's new procedures check out fine. All the materials are ready. I will do a few more pretests but I expect to make few if any changes. Recruiting will begin in about 10 days and the actual testing should start soon after the students return in February.

IV. APPENDIX: EXPERIMENTAL MATERIALS

GENERAL INSTRUCTIONS

You are participating in an experiment to test two similar computer systems. Both systems are used to edit text but are different in a number of ways. We are trying to determine the effects of these differences. To help us do this we ask that you do the following:

1) Today you will read a manual describing the use of one of the editing systems. Study the manual until you are completely satisfied that you can use the system to actually do some editing. You may take as long as you want but the session is timed so please go as fast as you can.

2) When you are ready we will give you an editing task which you will perform at the computer terminal. This will also be timed so please work as effectively as possible.

3) Before you leave today we will set up another appointment for about a week from now. When you return you will be given a brief quiz and will do another editing task. In the interim do not discuss the experiment with other participants. When you complete this second task you will receive $5 for your participation in this experiment.

When you are ready you will begin the timing by typing "START." The experimentor will give you the manual. IMPORTANT: The practice file is named "BETA."

When you are completely satisfied that you understand the system, end the training period by using the normal request to end an editing session (as described in the manual).

INSTRUCTIONS FOR TASK

Now we would like you to use the system you have just studied to actually perform an editing task. The procedures you will use are illustrated by this example:

1. You begin by typing START:

```
$$YOU MAY BEGIN WHEN READY
  start
```

2. You must name the file:

```
**WELCOME TO THE EDITING SESSION  JANUARY 02, 1978  4:59 PM
--edit,alpha
**THE CURRENT LINE IS
  And they set me on the doorstep. Oh, the
```

3. You have been given three printed pages. The first is the current state of the file and looks like this:

WHAT THE FILE LOOKS LIKE NOW

```
And they set me on the doorstep. Oh, the
Night was dark and wild
But when they lit the fuse, then I Jumped!
```

The second page shows what the file will look like when you have finished:

WHAT THE FILE SHOULD LOOK LIKE

```
And they set me on the doorstep. Oh, the
Night was dark and wild;
But when they lit the candle, then I
Smiled!
```

The last page is a listing showing the necessary editing:

REQUIRED CHANGES

```
And they set me on the doorstep. Oh, the

Night was dark and wild!
But when they lit the ~~fuse,~~ candle then I ~~jumped!~~
Smiled!
```

Notice that this last page is double spaced for legibility. The actual file, as shown on the other pages, is single spaced. You must perform the necessary editing:

```
--l;*
  And they set me on the doorstep. Oh, the
  Night was dark and wild
  But when they lit the fuse, then I Jumped!
--c
**ENTER NEW TEXT

++But when they lit the candle, then I
++Smiled!
++

**THE CURRENT LINE IS
  Smiled!
```

4. When all the necessary changes have been made you must save the file. If it does not match the goal text, the system will report the first line that fails to match (but only the first line, even if there are more). If this is the case you must continue editing:

```
--replace
**CURRENT FILE DOES NOT MATCH THE TARGET FILE
**THE FIRST LINE THAT DOES NOT MATCH IS
  Night was dark and wild
**PLEASE CONTINUE EDITING
**THE CURRENT LINE IS
  Smiled!
--f:/Night/;-1
  Night was dark and wild
--rs:/wild/,/wild;/
**THE CURRENT LINE IS
  Night was dark and wild;
```

If the files match, the system will report your success and you should end the session:

```
--replace
**CURRENT FILE MATCHES THE TARGET FILE
**YOU MAY STOP WHEN READY
--end
**END OF EDITING SESSION  5:01 PM
```

If this procedure is clear you are ready to begin the task. During the task you will have these materials:

1. The manual
2. What the file looks like now
3. What the file should look like
4. A listing of the required changes

When you are ready to begin please inform the experimentor.

INSTRUCTIONS FOR TASK

Now we would like you to use the system you have just studied to actually perform an editing task. The procedures you will use are illustrated by this example:

1. You begin by typing START:

```
$$YOU MAY BEGIN WHEN READY
  start
```

2. You must name the file:

```
**WELCOME TO THE EDITING SESSION  JANUARY 02, 1978  5:04 PM
--afi alpha
**THE CURRENT LINE IS
  And they set me on the doorstep. Oh, the
```

3. You have been given three printed pages. The first is the current state of the file and looks like this:

WHAT THE FILE LOOKS LIKE NOW

```
And they set me on the doorstep. Oh, the
Night was dark and wild
But when they lit the fuse, then I Jumped!
```

The second page shows what the file will look like when you have finished:

WHAT THE FILE SHOULD LOOK LIKE

```
And they set me on the doorstep. Oh, the
Night was dark and wild;
But when they lit the candle, then I
Smiled!
```

The last page is a listing showing the necessary editing:

REQUIRED CHANGES

```
And they set me on the doorstep. Oh, the

Night was dark and wild;
                        candle
But when they lit the ~~fuse~~, then I ~~jumped!~~
Smiled!
```

Notice that this last page is double spaced for legibility. The actual file, as shown on the other pages, is single spaced. You must perform the necessary editing:

```
--1a1
  And they set me on the doorstep, Oh, the
  Night was dark and wild
  But when they lit the fuse, then I Jumped!
--ito
**ENTER NEW TEXT

++But when they lit the candle, then I
++Smiled!
++

**THE CURRENT LINE IS
  Smiled!
```

4. When all the necessary changes have been made you must save the file. If it does not match the goal text, the system will report the first line that fails to match (but only the first line, even if there are more). If this is the case you must continue editing:

```
--p
**CURRENT FILE DOES NOT MATCH THE TARGET FILE
**THE FIRST LINE THAT DOES NOT MATCH IS
  Night was dark and wild
**PLEASE CONTINUE EDITING
**THE CURRENT LINE IS
  Smiled!
--bw/Night/
  Night was dark and wild
--c/wild/t/wild;/
**THE CURRENT LINE IS
  Night was dark and wild;
```

If the files match, the system will report your success and you should end the session:

```
--p
**CURRENT FILE MATCHES THE TARGET FILE
**YOU MAY STOP WHEN READY
--ee
**END OF EDITING SESSION  5:06 PM
```

If this procedure is clear you are ready to begin the task. During the task you will have these materials:

1. The manual
2. What the file looks like now
3. What the file should look like
4. A listing of the required changes

When you are ready to begin please inform the experimentor.

BEGINNING TEXT

One of the greatest mayors New York ever had was Fiorello La Gardia -- "The Little Flower." Every New Yorker remembers a strike kept the Sunday Journals off the stands. They remember, too, his squeaky fulminations against the "crooks" and "tinhorns" in our town, and his weekly radio sign off, "Patience and fortitude".

Once the mayor was presiding over night court. A man was brought before him, charged with stealing a loaf of bread. He claimed that his family was starving.

"I must punish you," declared La Gardia. "The law makes no exceptions. I must fine you ten dollars." But The Little Flower was reaching into his own pocket and remembering his own early life of frequent deprivation as he added, "Well, here's the ten dollars to pay your fine -- which I now remit." He tossed the ten dollars into his famous hat. "Furthermore," he declared, "I'm going to fine everybody in this courtroom fifty cents for living in a city that can't provide for its own citizens and forces decent men to steal in order to feed their families."

The hat was passed and an incredulous old man, with a renewed faith in mankind, left the courtroom with a stake of $47.50. No one in the room felt that Justice had ever been better served.

TARGET TEXT

One of the greatest mayors New York ever had was Fiorello La Guardia -- "The Little Flower." Every New Yorker remembers the day Fiorello read the funny papers over the radio -- with all the appropriate excitement and inflections -- when a strike kept the Sunday Journals off the stands. They remember, too, his squeaky fulminations against the "crooks" and "tinhorns" in our town, and his weekly radio sign off, "Patience and fortitude."

One time the ubiquitous mayor chose to preside in a night court. It was bitter cold outside. A trembling man was brought before him, charged with stealing a loaf of bread. His family, he said, was starving.

"I've got to punish you," declared La Guardia. "The law makes no exceptions. I must fine you ten dollars." But The Little Flower was reaching into his own pocket as he added, "Well, here's the ten dollars to pay your fine -- which I now remit." He tossed the ten dollars into his famous sombrero. "Furthermore," he declared, "I'm going to fine everybody in this courtroom fifty cents for living in a town where a man has to steal bread in order to eat. Mr. Bailiff, collect the fines and give them to this defendant!"

The hat was passed and an incredulous old man, with a light of heaven in his eyes, left the courtroom with a stake of $47.50.

REQUIRED EDITING

One of the greatest mayors New York ever had was Fiorello La Gardia -- "The Little Flower." Every New Yorker remembers the day Fiorello read the ~~funnies~~ funny papers over the radio -- a strike kept the Sunday journals off the stands. They remember, too, his squeaky fulminations against the "crooks" and "tinhorns" in our town, and his weekly radio sign off, with all the appropriate excitement and inflections -- when "Patience and fortitude".

~~Once the mayor was presiding over night court. A man~~ One time the ubiquitous mayor chose to preside in a night court. It was bitter cold outside. A trembling man was brought before him, charged with stealing a loaf of bread. ~~He claimed that his family was starving.~~ His family, he said, was starving.

"I must punish you," declared La Gardia. "The law makes no exceptions. I must fine you ten dollars." But The Little Flower was reaching into his own pocket ~~and~~ as he ~~remembering his own early life of frequent deprivation as he~~ added, "Well, here's the ten dollars to pay your fine -- which I now remit." He tossed the ten dollars into his famous ~~hat~~ sombrero. "Furthermore," he declared, "I'm going to fine everybody in this courtroom fifty cents for living in a ~~city that can't provide for its own citizens and forces decent men to steal in order to feed their families.~~ town where a man has to steal bread in order to eat. MR. Bailiff, collect the fines and give them to this defendant!"

The hat was passed and an incredulous old man, with a ~~renewed faith in mankind,~~ light of heaven in his eyes, left the courtroom with a stake of $47.50. ~~No one in the room felt that justice had ever been better served.~~ No one in the room felt that justice had ever

SECOND SESSION INSTRUCTIONS

A week ago you studied a manual which described a text editing system. You then used this system to perform an actual editing task. Today you will perform a similar task. The procedure will be as before except that there will be no training period. You will be allowed to use the manual and you may review the task instructions before you begin. First, however, please answer the questions on the next three pages. When you have finished signal the experimentor.

RECALL QUIZ

INSTRUCTIONS: We are interested in your recall of the editing requests you used in last week's session. For each of the editing situations described below please write a request (or requests) which will perform the needed editing. Write the requests just as you would type them on the computer terminal.

1. Advance the line pointer 5 lines.

2. The phrase:
   ONLY ONE
is in the current line. Replace it with the phrase:
   AT LEAST TWO

3. List 5 lines beginning at the current line.

4. The current line reads:
   THIS IS A TEXT EDITING SYSTEM
Make it read:
   THE SYSTEM IS USED TO DO EDITING

5. Move the current line pointer forward to the second line containing the phrase:
   EDITING SYSTEM

6. Remove the boundary limiting the scope of editing.

7. Move the 3 lines beginning with the current line into the holding file.

8. Delete the next line of text containing the phrase:
   COMPUTER TERMINAL

9. Move the line pointer to the beginning of the file.

10. The file to be edited is called:
GREEN
Name the file as if you were starting an editing session.

SUBJECTIVE IMPRESSIONS

INSTRUCTIONS: We would like to determine your subjective reactions to the system you used last week. Please use a scale of 1 to 5 in answering the questions below.

1. Was the manual easy to understand?

EASY				HARD
1	2	3	4	5

2. Were the editing requests easy to use?

EASY				HARD
1	2	3	4	5

3. When using the system did you feel that you made many errors?

FEW				MANY
1	2	3	4	5

4. How often did you refer to the manual when using the system?

OFTEN				RARELY/NEVER
1	2	3	4	5

5. As you did the task did you feel that you quickly became comfortable with the system?

QUICKLY				SLOWLY
1	2	3	4	5

6. Overall were you satisfied with the system as a means of editing text?

VERY SATISFIED				VERY DISSATISFIED
1	2	3	4	5

7. Did you enjoy the text editing session?

ENJOYED				DID NOT ENJOY
1	2	3	4	5

Finally, please answer these three brief "essay" questions:

8. Which commands were easiest to understand and use? Which were the hardest? Why?

9. Are there any requests that could be added to improve the efficiency of the system?

10. What changes could be made to improve the structure of the editing request language?

EXPERIMENTOR'S FORM

1. (Ask the subject)
Are you at all familiar with the NOS Text Editor? This is the one that is called by the command "EDIT" and should not be confused with the UMASS Sequenced Text Editor which is part of the FORTRAN System:

Yes (if yes, drop) No

2. (Ask the subject, reading alternatives)
How many hours would you say that you have used a computer terminal?

0-5 6-50 OVER 50

3. Do you consider yourself a professional typist?

Yes No

4. We would like to include you in an experiment to test a new computer system. The experiment will involve two meetings a week apart and will take a total of about two or three hours. You will be paid an incentive of $5 for your participation. Are you interested? (If yes, go on) We will need to be able to contact you to set up an appointment. How can we reach you?

TEL. NO. _____

When would be the best times for us to try to set up an appointment for you?

ASSIGNED TO LEVEL:

APPOINTMENT FOR FIRST SESSION:

APPOINTMENT FOR SECOND SESSION:

I have received an incentive payment of $5 for my participation in this experiment.

DATE: SIGNATURE:

SECOND SESSION REMINDER

Appointment in GRES A303E

on:

at:

RS TEXT EDITING SYSTEM

This user's guide will describe a time sharing system that allows the user to modify text using a small set of requests. It is divided into three sections:

1) General Concepts

2) Editing Requests

3) Sample Editing Session

SECTION 1: GENERAL CONCEPTS

TERMINAL USAGE

Text editing is a process of creating, maintaining, and updating files of text. The editing requests described here make it possible to insert, delete, or substitute text. You will issue these requests while working at a time sharing terminal. There are a few ground rules about using a terminal that you should know:

1) The backspace is used to delete a character. Thus, if you mistype a character you may backspace and strike over to correct.

2) Depressing ESC causes the whole line you are typing to be deleted.

3) When the line (either a request or text input) is finished, depress RETURN and the operation will be performed.

4) The interrupt facility is used to halt certain requests. To use the interrupt facility press the CTRL key and, while pressed, type the letter "C".

5) When the printing element of the terminal is not typing it moves 3 spaces to the right so that you can read what you have typed. Keep this in mind if you need exact allignment of characters.

EDITING CONCEPTS

It is also necessary to explain a few of the concepts related to editing. These are:

FILES:

A file is a named collection of information that the editor maintains for you. When a file is created it is given a name. From then on both you and the editor refer to the file by that name.

LINES:

A file is made up of lines. Each line may be up to 150 characters long. If you enter a line that is shorter than this (as you almost always will) the system will fill in the remaining spaces with blanks. For example, if you wanted to type a line of all blanks you would only need to type one blank (by depressing the "space" bar) and then depress "RETURN."

THE CURRENT LINE:

Editing requests can be used to effect changes in the file you are editing. Some requests apply to the file as a whole: for example, you may want to change every instance of the name "SMITH" to "JONES." Other requests may apply only to one specific line in the file: you might want to delete one line. For this reason the editor maintains a pointer to one line in the file. This is called the "current line." Initially, the current line is the first line of the file. Thereafter, the last line printed is the current line.

PROMPTING:

Two prompting characters are printed by the editor whenever it is ready to receive something from you. The characters indicate what type of response is expected from you.

Prompting Sequence	Response Type
--	Requests
++	Input of New Lines
//	Answers to Questions

The last type of prompt, "//", requires you to type YES or NO (Y or N will do).

GRAMMATICAL NOTATION:

Every language has a grammar, and the editor's request language is no exception. In the descriptions that follow we employ a special notation to describe the grammatical form of each request. The rules for this notation are as follows:

1) KEYWORDS: Words shown in upper case are keywords. A keyword introduces a request; other keywords qualify a request. Most keywords may be abbreviated by their first letter (exceptions to this rule will be noted below). If not abbreviated, a keyword must be spelled out correctly. Thus, the two commands below are equivalent:

```
FIND:/RELAR/;2          F:/RELAR/;2
```

Also, note that inserting spaces within requests has no effect. The two commands below are equivalent:

```
FIND:/RELAR/;2          F    :   /RELAR/ ;  2
```

2) OBJECTS: Words shown in lower case are names for the objects of a request that you supply to make the request. These words may specify the name of a file or piece of text.

3) ALTERNATIVES: Keywords or objects that are grouped together and separated by the character "|" are mutually exclusive alternatives.

4) OPTIONS: Keywords, objects, or any groupings of these that are in parentheses represent parts of a request whose use is optional.

Thus for example:

LIST (n LINES | ALL LINES) (WITH /text/)

means that any of the commands below would be valid:

L3LW/anyword/

LALW/anyword/

L3L

LAL

LW/anyword/

L

SECTION 2: EDITING REQUESTS

SUMMARY:

```
FORWARD      (n LINES | ALL LINES)      (WITH /text/)
BACKWARD     (n LINES | ALL LINES)      (WITH /text/)
LIST         (n LINES | ALL LINES)      (WITH /text/)
DELETE       (n LINES | ALL LINES)      (WITH /text/)
HOLD         (n LINES | ALL LINES)      (WITH /text/)

INSERT TEXT AFTER  }
INSERT TEXT OVER   }   After prompt, type new lines.
INSERT BUFFER AFTER
INSERT BUFFER OVER

CHANGE  (n | ALL)  /old-text/ TO /new-text/

ASSUME FILE IS file-name
PRESERVE
END EDITING
```

DETAILED DESCRIPTIONS OF REQUESTS:

BEGINNING THE EDITING SESSION:

In order to start editing you must name the file to be edited by saying:

ASSUME FILE IS file-name

(Note: The computer may take a long time to do this request.)

SPECIFYING TEXT:

Before describing each request individually it is necessary to explain the general method of specifying what text is to be operated upon. Except for the CHANGE request, all requests operate on a whole line or lines. There are two ways of specifying which line you want:

1) You may supply only a number. Thus:

LIST 3 LINES

will list the 3 lines beginning with the "current line." If the number and the word "LINES" are omitted one line is assumed.

2) You may supply a piece of text and a number. Thus:

LIST 3 LINES WITH /text/

will list the next 3 lines containing the given "text." The search for these lines starts at and includes the "current line." Again, if the number and word "LINES" are omitted one line is assumed..

The given "text" is any actual sequence of characters being specified, with blanks treated like other characters. Notice that the character "/" is used to bracket the actual character sequence. Since this character indicates both the beginning and end of the desired text, it must not appear in the text itself.

When referencing a long piece of text, it is tiresome to have to type it all out when only a few details will identify

it uniquely. The editor allows you to specify long text strings by their beginning and end, replacing the middle piece of text with the ellipsis symbol " ... ". Thus:

/THE...FORE/

would refer to any piece of text starting with "THE" and ending with "FORE." Some caution must be used with this since, continuing the example above, the specifed string could be either "THE GOLFER YELLED FORE" or "THEREFORE."

MOVING THE LINE POINTER:

The commands FORWARD and BACKWARD are used to move the line pointer. They also cause lines to be printed. (Remember that the "current line" is always the last line to be printed.) There are several ways to use these commands:

1) FORWARD or BACKWARD

The editor moves the pointer ahead (back) one line.

2) FORWARD n LINES or BACKWARD n LINES

The editor moves the pointer ahead (back) n lines (where n is a number).

3) FORWARD ALL or BACKWARD ALL

The editor moves the pointer ahead to the last line of the file (back to the first line).

4) FORWARD WITH /text/ or BACKWARD WITH /text/

The editor moves the pointer ahead (back) to the next line containing an occurrence of the specified text. The search excludes the current line.

5) FORWARD n LINES WITH /text/

or BACKWARD n LINES WITH /text/

The editor moves the pointer ahead (back) to the nth line containing an occurrence of the specified text.

6) FORWARD ALL LINES WITH /text/

or BACKWARD ALL LINES WITH /text/

The editor moves the pointer ahead (back) to the last (first) line containing an occurrence of the specified text.

DISPLAYING LINES:

To display one or more lines of text, just say:

LIST (n LINES | ALL LINES) (WITH /text/)

The variations on the list request are similar to the FIND request:

1) LIST

Only the current line is displayed on the terminal.

2) LIST n LINES
LIST ALL LINES

The editor displays the next n or ALL lines including the current line.

3) LIST n LINES WITH /text/
LIST ALL LINES WITH /text/

The next n (or ALL) lines containing the specified text are displayed. Remember that the last line printed is the current line, thus the LIST request may cause the current line to be changed.

DELETING LINES:

In order to delete one or more lines of text you say:

DELETE (n LINES | ALL LINES) (WITH /text/)

This operation is virtually identical to the LIST request with the difference that the particular lines specified are not displayed but removed from the file.

MOVING LINES:

When editing a file with misplaced lines it may be convenient to move the lines temporarily into a holding file. The original lines can then be deleted and the copy inserted elsewhere in the text. To move one or more lines into the holding file say:

HOLD (n LINES | ALL LINES) (WITH /text/)

This request erases the previous contents of the holding file and replaces them with the specified lines. The moved lines are not erased from the original file.

ADDING LINES:

To add one or more lines to the file say:

INSERT TEXT AFTER

The editor will then prompt you for a line to add. Type in the line and depress RETURN . The editor will prompt for more input. You may add as many lines as you want in this manner. To stop adding lines and go on to the next request, use the interrupt (CTRL-C). The line(s) added will be placed after the "current line."

CHANGING LINES:

To replace a whole line with one or more lines, say:

INSERT TEXT OVER

The editor will then prompt you for a line. Continue as above, using CTRL-C to indicate when you are through. The new line(s) will replace the "current line."

INSERTING THE CONTENTS OF THE HOLDING FILE:

The request TRANSFER allowed you to move one or more lines from the file being edited into a holding file. To add the contents

of the holding file to the file being edited say:

INSERT BUFFER AFTER

This will place the contents of the holding file directly after the current line. If you say:

INSERT BUFFER OVER

the contents of the holding file will replace the current line.

CHANGING TEXT WITHIN A LINE:

In order to change a piece of text in one or more lines say:

CHANGE (n | ALL) /old-text/ TO /new-text/

This request is different from all the other requests in that it operates on text within lines rather than whole lines themselves. Starting within the current line, the next n (or ALL) occurrences of the old text given are replaced by the new text given.

If no new text is given between the second pair of brackets each occurrence of text will be deleted. You may not use the ellipsis symbol method of specifying long text strings when using this command.

THE CURRENT LINE REVISITED:

You are reminded once again about the current line rules: 1) When a FIND or LIST command is executed the last line printed is the current line. 2) All other requests cause the message:

THE CURRENT LINE IS

followed by the current line, whether changed or not. 3) An unsuccessful request leaves the current line unchanged. (A request will be unsuccessful if it contains a grammatical error or if the operation can't be performed. For example, if a file is only 10 lines long, the command

could not be done and would be unsuccessful.)

ENDING THE EDITING SESSION:

When you have completed your editing you must save the newly edited file by saying:

PRESERVE

After your edited file has been saved you will end the editing session by saying:

END EDITING

IMPORTANT: Do not use END before saving the file or the new file will be lost.

SAMPLE EDITING SESSION

Note that lines preceeded by ** are messages from the editor to you. Lines preceeded by -- (which the editor types as a prompt) are typed by you. Lines preceeded by no special characters are the text in the file, as reported by the editor.

```
**WELCOME TO THE EDITING SESSION  JANUARY 02, 1978  3:37 PM
--ASSUME FILE IS ALPHA                                    +++  You must first name the file. The editor sets the current line at the
**THE CURRENT LINE IS                                          beginning of the file.
  What am I?
--LIST ALL LINES                                          +++  You ask to display the whole file.
  What am I?
  They chose me from my brothers: 'That's the
  actual number of lines
  Nicest one,' they said,
  Candle in my head;
  And they carved me out a face and put a
  Night was dark and will
  But when they lit the fuse, then I JUMPED!                   After the list, the line pointer will be at the end of the file (the
--BACKWARD /actual/                                       +++  last line printed). You try to move it to the line containing the word
**        /                                                    "actual" but leave out the keyword WITH.
**INVALID CHARACTERS AT END OF BACKWARD REQUEST
--bACKwaRD WIth /actual/                                  +++  This time you succeed. Notice that you can mix upper and lower case in
  actual number of lines                                       keywords.
--delete 1 line                                           +++  You try to delete the current line but leave out the "S" in LINES.
**       /                                                     Keywords must be spelled exactly.
**INVALID CHARACTERS AT END OF DELETE REQUEST
--d1 lines                                                +++  This time it works. Notice that keywords can be abbreviated by their
**DELETE 1 LINE(S)                                             first letter and spaces between keywords aren't necessary.
**THE CURRENT LINE IS
  Nicest one,' they said,
--f                                                       +++  You advance the line pointer one line (if no number is mentioned, one is
  Candle in my head;                                           assumed).
--hold                                                    +++  You move the current line into the holding file ...
**HOLD 1 LINE(S) INTO BUFFER
**THE CURRENT LINE IS
  Candle in my head;
--d                                                       +++  Now delete the current line ...
**DELETE 1 LINE(S)
**THE CURRENT LINE IS
  And they carved me out a face and put a
--insert buffer after                                     +++  And insert the holding file. Thus, you have moved the desired line from
**INSERT BUFFER AFTER CURRENT LINE                             one position to another.
**THE CURRENT LINE IS
  Candle in my head;
--insert text after                                       +++  Use this command to insert new text.
**ENTER NEW TEXT                                               The editor prompts for a line.
                                                          +++  You enter a line in response to the prompt.
++And they set me on the doorstep. Oh, the                +++  By using the interrupt facility (CTRL-C) you end the insertion.
++

**THE CURRENT LINE IS
  And they set me on the doorstep. Oh, the
--f
  Night was dark and will
--change /will/ to /wild/                                 +++  Here you use CHANGE to modify text within a line, replacing "will"
**THE CURRENT LINE IS                                          with "wild."
  Night was dark and wild
--
```

```
--rs:/will/,/wild/                                  +++  Here you use RS to modify text within a line, replacing "will" with
**THE CURRENT LINE IS                                    "wild."
  Night was dark and wild
--chan                                              +++  You start the next operation but notice you must first move the current
**RE-ENTER THE LAST LINE                                 line pointer.  You type ESC instead of RETURN and the editor asks you
--f:/But/,/Fuse/                                    +++  to re-enter the line.
  But when they lit the fuse, then I Jumped!        +++  You move the current line pointer.  Notice the use of the ellipsis feature.
--change
**ENTER NEW TEXT                                    +++  Now you use this command to change a whole line.

++But when they lit the candle, then I              +++  This line replaces the old line.
++Smiled!                                           +++  This line is added after the one above.
++                                                  +++  You press CTRL-C to end the insertion.  Two lines replace the original line.

**THE CURRENT LINE IS
  Smiled!
--f:-x                                              +++  To check your work you move the current line pointer to the beginning of
  What am I?                                             the file ...
--l:*                                               +++  And list the whole file.
  What am I?
  They chose me from my brothers: 'That's the
  Nicest one,' they said,
  And they carved me out a face and put a
  Candle in my head;
  And they set me on the doorstep. Oh, the
  Night was dark and wild
  But when they lit the candle, then I
  Smiled!
--replace                                           +++  It looks OK so you save the new file.
--end                                               +++  And end the editing session.
**END OF EDITING SESSION  4:06 PM
```

OS TEXT EDITING SYSTEM

This user's guide will describe a time sharing system that allows the user to modify text using a small set of requests. It is divided into three sections:

1) General Concepts

2) Editing Requests

3) Sample Editing Session

SECTION 1: GENERAL CONCEPTS

TERMINAL USAGE

Text editing is a process of creating, maintaining, and updating files of text. The editing requests described here make it possible to insert, delete, or substitute text. You will issue these requests while working at a time sharing terminal. There are a few ground rules about using a terminal that you should know:

1) The backspace is used to delete a character. Thus, if you mistype a character you may backspace and strike over to correct.

2) Depressing ESC causes the whole line you are typing to be deleted.

3) When the line (either a request or text input) is finished, depress RETURN and the operation will be performed.

4) The interrupt facility is used to halt certain requests. To use the interrupt facility press the CTRL key and, while pressed, type the letter "C".

5) When the printing element of the terminal is not typing it moves 3 spaces to the right so that you can read what you have typed. Keep this in mind if you need exact allignment of characters.

EDITING CONCEPTS

It is also necessary to explain a few of the concepts related to editing. These are:

FILES:

A file is a named collection of information that the editor maintains for you. When a file is created it is given a name. From then on both you and the editor refer to the file by that name.

LINES:

A file is made up of lines. Each line may be up to 150 characters long. If you enter a line that is shorter than this (as you almost always will) the system will fill in the remaining spaces with blanks. For example, if you wanted to type a line of all blanks you would only need to type one blank (by depressing the "space" bar) and then depress "RETURN."

THE CURRENT LINE:

Editing requests can be used to effect changes in the file you are editing. Some requests apply to the file as a whole: for example, you may want to change every instance of the name "SMITH" to "JONES." Other requests may apply only to one specific line in the file: you might want to delete one line. For this reason the editor maintains a pointer to one line in the file. This is called the "current line." Initially, the current line is the first line of the file. Thereafter, the last line printed is the current line.

PROMPTING:

Two prompting characters are printed by the editor whenever it is ready to receive something from you. The characters indicate what type of response is expected from you.

Prompting Sequence	Response Type
--	Requests
++	Input of New Lines
//	Responses to Questions

The last type of prompt, "//", requires you to type YES or NO (Y or N will do).

GRAMMATICAL NOTATION:

Every language has a grammar, and the editor's request language is no exception. In the descriptions that follow we employ a special notation to describe the grammatical form of each request. The rules for this notation are as follows:

1) KEYWORDS: Words shown in upper case are keywords. A keyword introduces a request; other keywords qualify a request. Any keyword may be abbreviated by its first letter. If not abbreviated, a keyword must be spelled out correctly. For example the two commands below are equivalent:

FORWARD 2 LINES WITH /RELAR/ F 2 L W /RELAR/

Also, note that inserting spaces within requests has no effect. The two commands below are equivalent:

F 2 L W /RELAR/ F2LW/RELAR/

2) OBJECTS: Words shown in lower case are names for the objects of a request that you supply to make the request. These words may specify the name of a file or piece of text.

3) ALTERNATIVES: Keywords or objects that are grouped together and separated by the character "|" are mutually exclusive alternatives.

4) OPTIONS: Keywords, objects, or any groupings of these that are in parentheses represent parts of a request whose use is optional.

Thus for example:

LIST (:/text/) (#n | #*)

means that any of the commands below would be valid:

L:/anyword/#3

L:/anyword/#*

L:/anyword/

L#3

L#*

L

SECTION 2: EDITING REQUESTS

SUMMARY:

```
FIND      (:/text/)   (;n | ;*)
FIND      (:/text/)   (;-n | ;-*)
LIST      (:/text/)   (;n | ;*)
DELETE    (:/text/)   (;n | ;*)
EXTRACT   (:/text/)   (;n | ;*)

ADD       }
CHANGE    }   After prompt, type new lines.
ADD       }
CHANGE    }   After prompt, type $.

RS:  /old-text/,/new-text/  (;n | ;*)

EDIT,file-name
REPLACE
END
```

NOTE: The keywords RS, EDIT, REPLACE, and END cannot be abbreviated.

DETAILED DESCRIPTIONS OF REQUESTS:

BEGINNING THE EDITING SESSION:

In order to start editing you must name the file to be edited by saying:

EDIT,file-name

(Note: The computer may take a long time to do this request.)

SPECIFYING TEXT:

Before describing each request individually it is necessary to explain the general method of specifying what text is to be operated upon. Except for the RS request, all requests operate on a whole line or lines. There are two ways of specifying which line you want:

1) You may supply only a number. Thus:

LIST;3

will list the 3 lines beginning with the "current line." If the semi-colon and number are omitted one line is assumed.

2) You may supply a piece of text and a number. Thus:

LIST:/text/;3

will list the next 3 lines containing the given "text." The search for these lines starts at and includes the "current line." Again, if the semi-colon and number are omitted one line is assumed.

The given "text" is any actual sequence of characters being specified, with blanks treated like other characters. Notice that the character "/" is used to bracket the actual character sequence. Since this character indicates both the beginning and end of the desired text, it must not appear in the text itself.

When referencing a long piece of text, it is tiresome to have to type it all out when only a few details will identify it uniquely. The editor allows you to specify long text strings

by their beginning and end, omitting the middle piece of text. To do this replace:

/text/

with:

/start-text/,/end-text/

Thus:

/THE/,/FORE/

would refer to any piece of text starting with "THE" and ending with "FORE." Some caution must be used with this since, continuing the example above, the specifed string could be either "THE GOLFER YELLED FORE" or "THEREFORE."

MOVING THE LINE POINTER:

The command FIND is used to move the line pointer. It also causes lines to be printed. (Remember that the "current line" is always the last line to be printed.) There are several ways to use this command:

1) FIND or FIND;-1

The editor moves the pointer ahead (back) one line.

2) FIND;n or FIND;-n

The editor moves the pointer ahead (back) n lines (where n is a number).

3) FIND;* or FIND;-*

The editor moves the pointer ahead to the last line of the file (back to the first line).

4) FIND:/text/ or FIND:/text/;-1

The editor moves the pointer ahead (back) to the next line containing an occurrence of the specified text. The search excludes the current line.

5) FIND:/text/;n or FIND:/text/;-n

The editor moves the pointer ahead (back) to the nth line containing an occurrence of the specified text.

6) FIND:/text/;* or FIND:/text/;-*

The editor moves the pointer ahead (back) to the last (first) line containins an occurrence of the specified text.

DISPLAYING LINES:

To display one or more lines of text, just say:

LIST (:/text/) (;n | ;*)

The variations on the list request are similar to the FIND request:

1) LIST

Only the current line is displayed on the terminal.

2) LIST;n
LIST;*

The editor displays the next n (or all if * is used) lines including the current line.

3) LIST:/text/;n
LIST:/text/;*

The next n (or all) lines containins the specified text are displayed. Remember that the last line printed is the current line. Thus the LIST request may cause the current line to be changed.

DELETING LINES:

In order to delete one or more lines of text you say:

DELETE (:/text/) (;n | ;*)

This operation is virtually identical to the LIST request with the difference that the particular lines specified are not displayed but removed from the file.

MOVING LINES:

When editing a file with misplaced lines it may be convenient to move the lines temporarily into a holding file. The original line can then be deleted and the copy inserted elsewhere in the text. To move one or more lines into the holding file say:

EXTRACT (:/text/) (.n | .*)

This request erases the previous contents of the holding file and replaces them with the specified lines. The moved lines are not erased from the original file.

ADDING LINES:

To add one or more lines to the file say:

ADD

The editor will then prompt you for a line to add. Type in the line and depress RETURN . The editor will prompt for more input. You may add as many lines as you want in this manner. To stop adding lines and go on to the next request, use the interrupt (CTRL-C). The line(s) added will be placed after the "current line."

CHANGING LINES:

To replace a whole line with one or more lines, say:

CHANGE

The editor will then prompt you for a line. Continue as above, using CTRL-C to indicate when you are through. The new line(s) will replace the "current line."

INSERTING THE CONTENTS OF THE HOLDING FILE:

The request EXTRACT allowed you to move one or more lines from the file being edited into a holding file. To add the contents of the holding file to the file being edited say:

ADD

When the editor prompts you for input, type "$". This will place the contents of the holding file directly after the current line. If you say:

CHANGE

and then type a "$" after the prompt, the contents of the holding file will replace the current line.

CHANGING TEXT WITHIN A LINE:

In order to change a piece of text in one or more lines say:

RS: /old-text/,/new-text/ (;n | ;*)

This request is different from all the other requests in that it operates on text within lines rather than whole lines themselves. Starting within the current line, the next n (or all) occurrences of the old text given are replaced by the new text given.

If no new text is given between the second pair of brackets each occurrence of text will be deleted. You may not use the

/start-text/,/end-text/

method of specifying long text strings when using this command. RS cannot be abbreviated.

THE CURRENT LINE REVISITED:

You are reminded once again about current line rules: 1) when a FIND or LIST command is executed the last line printed is the new current line. 2) All other requests cause the message:

THE CURRENT LINE IS

followed by the current line, whether changed or not. 3) An unsuccessful request leaves the current line unchanged. (A request will be unsuccessful if it contains a grammatical error or if the operation can't be performed. For example, if a file is only 10 lines long, the command:

FIND;20

could not be done and would be unsuccessful.)

ENDING THE EDITING SESSION:

When you have completed your editing you must save the newly edited file by saying:

REPLACE

After your edited file has been saved you will end the editing session by saying:

END

IMPORTANT: Do not use END before saving the file or the new file will be lost. The requests EDIT, REPLACE, and END must be spelled out in full and cannot be abbreviated.

SAMPLE EDITING SESSION

**Note that lines preceeded by ** are messages from the editor to you. Lines preceeded by --
(which the editor types as a prompt) are typed by you. Lines preceeded by no special characters
are the text in the file, as reported by the editor.**

```
**WELCOME TO THE EDITING SESSION  JANUARY 02, 1978  4:01 PM
--EDIT,ALPHA                                          + + +   You must first name the file.  The editor sets the current line at
**THE CURRENT LINE IS                                         the beginning of the file.
  What am I?
--LIST;*                                              + + +   You ask to display the whole file.
  What am I?
  They chose me from my brothers: 'That's the
  actual number of lines
  Nicest one,' they said,
  Candle in my head:
  And they carved me out a face and put a
  Night was dark and will
  Ink when they lit the fuse, then I jumped!
--find /actual/ ;-1                                   + + +   After the list, the line pointer will be at the end of the file (the
**                                                            last line printed).  You try to move it to the line containing the
**INVALID CHARACTERS AT END OF FIND REQUEST                    word "actual" but leave out the colon before the text string.
--FIND 1/actual/ ;-1                                  + + +   This time you succeed.  Notice that you can mix upper and lower case
  actual number of lines                                      in keywords.
--delet ;1                                            + + +   You try to delete the current line but leave out the final "E" in DELETE.
**/                                                           Keywords must be spelled exactly.
**INVALID CHARACTERS AT END OF DELETE REQUEST
--d;1                                                 + + +   This time it works.  Notice that keywords can be abbreviated by their first
**DELETE 1 LINE(S)                                            letter and spaces between keywords aren't necessary.
**THE CURRENT LINE IS
  Nicest one,' they said,                                     You advance the line pointer one line (if no number is mentioned, one is
--f                                                   + + +   assumed).
  Candle in my head;
--extract                                             + + +   You move the current line into the holding file ...
**EXTRACT 1 LINE(S) INTO BUFFER
**THE CURRENT LINE IS
  Candle in my head;
--d                                                   + + +   Now delete the current line ...
**DELETE 1 LINE(S)
**THE CURRENT LINE IS
  And they carved me out a face and put a
--add                                                 + + +   And insert the holding file by typing $ in response to the ENTER NEW TEXT
**ENTER NEW TEXT                                              prompt.  Thus, you have moved the desired line from one position to
                                                              another.
++$

**INSERT BUFFER AFTER CURRENT LINE
**THE CURRENT LINE IS
  Candle in my head;
--add
**ENTER NEW TEXT

++And they set me on the doorstep. Oh, the            + + +   You enter a line.
++                                                    + + +   By using the interrupt facility (CTRL-C) you end the insertion.

**THE CURRENT LINE IS
  And they set me on the doorstep. Oh, the
--f
  Night was dark and will
--
```

```
[illegible]
[illegible] THE LAST LINE
--twist...fuse/
  But when they lit the fuse, then I Jumped!
--insert text over
**ENTER NEW TEXT

++But when they lit the candle, then I
++Smiled!
++

**THE CURRENT LINE IS
  Smiled!
--bel
  What am I?
--lel
  What am I?
  They chose me from my brothers: 'That's the
  Nicest one,' they said,
  And they carved me out a face and put a
  Candle in my head;
  And they set me on the doorstep. Oh, the
  Night was dark and wild
  But when they lit the candle, then I
  Smiled!
--preserve
--end editing
**END OF EDITING SESSION  3:44 PM
```

+ + + You start the next operation but notice you must first move the current line pointer. You type ESC instead of RETURN and the editor asks you to re-enter the line.

+ + + You move the current line pointer. Notice the use of the ellipsis feature.

+ + + Now you use this command to change a <u>whole</u> line.

+ + + This line replaces the old line.

+ + + This line is added after the one above.

+ + + You press CTRL-C to end the insertion. Two lines replace the original line.

+ + + To check your work you move the current line pointer to the beginning of the file ...

+ + + And list the whole file.

+ + + It looks OK so you save the new file.

+ + + And end the editing session.