READIT! A Text Presentation Application for the Macintosh

Users' Manual

Elizabeth Saul, Mike Pohl, and Susan R. Goldman
University of California, Santa Barbara

Technical Report
December, 1988

The software described in this report was developed with funding from the Cognitive Science Program, Office of Naval Research, under Contract N00014-85-K-0562, authorization number NR442c015. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION University of California Santa Barbara 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Cognitive Science Program Office of Naval Research (Code 1142CS)
6c. ADDRESS (City, State, and ZIP Code) Department of Education University of California Santa Barbara, CA 93106	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-K0562
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Readit! A Text Presentation Application for the MacIntosh Users' Manual

12. PERSONAL AUTHOR(S)
Elizabeth Saul, Mike Pohl, and Susan R. Goldman

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 1985 TO 1988	1988, December, 28	37

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
05	10		Text Comprehension; Computer Presentation of Text

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

READIT! is a text presentation application for use on the MacIntosh computer. It presents a pre-specified segment of text in the window and records the amount of time spent on the window. Progress through the text consists of viewing a sequence of windows, each displaying one segment of text. Segments may be viewed in a forward direction or in a backward direction. A complete trace of the readers progress through the text if kept. From that trace, we can reconstruct the order in which segments were read and how much time was spent reading each. A series of data treatment applications perform compute rate and process time measures and prepare data tables from which various summary statistics may be computed. In addition to READIT!, the MacWrite or MSWord word processing applications are necessary. (JES)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Susan Chipman	22b. TELEPHONE (Include Area Code) 202-696-4318	22c. OFFICE SYMBOL ONR1142CS

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Table of Contents

ABSTRACT

READIT! is a text presentation application for use on the MacIntosh computer. It presents a pre-specified segment of text in the window and records the amount of time spent on the window. Progress through the text consists of viewing a sequence of windows, each displaying one segment of text. Segments may be viewed in a forwaro direction or in a backward direction. A complete trace of the readers progress through the text if kept. From that trace, we can reconstruct the order in which segments were read and how much time was spent reading each. A series of data treatment applications perform compute rate and process time measures and prepare data tables from which various summary statistics may be computed. In addition to READIT!, the MacWrite or MSWord word processing applications are necessary.

SUMMARY OF READIT!

READIT! is a series of MacIntosh applications designed to present text, record reading time per window, record the sequence in which windows are viewed, and prepare the reading time data for subsequent statistical analyses. Our use of the package involved presenting one sentence per window but other "per window" segments are possible. Readers can skim forward and backward through the segments much the way they might when reading normally.

The software was developed for a Macintosh II computer with or without a hard drive, and with the extended keyboard. The software will work on any Macintosh with 512K RAM, including Mac SE and MacII. The software is written in MPW Pascal and cannot be translated to other machine types because the resource code is Mac dedicated. A program disk and complete listings of the Resource Code and of the Source Code are available upon request from the third author.

The data collected and manipulated by these programs consist of reading rate, processing time, and reading sequence for sentences and passages. Ceilings on the number of passages and segments permissible within a given testfile make research into smaller units of text, such as words, unfeasible without modifications to the program. In addition, this package is **not** intended to collect data on eye movement, vocabulary comprehension, or perceptual dysfunctions.

The following overview briefly describes the function of each application and where in the research process each application falls. More complete information regarding the purpose, constraints, use, and output of the applications is contained in the chapter for each.

1. Macwrite/Ms Word

The texts selected for subjects to view are typed in one of these word processors. In order to gather chronometric data on a meaningful unit of text, it is decided ahead of time how much of each text will be shown on each screen view; the passages are divided into these "segments" by delimiting them. Once typed, corrected, and delimited, each passage is saved as a unique "text only" file.

2. Buildfile

The delimited passages created above are prepared for computer use by the Buildfile application. Specifically, Buildfile breaks each passage down into its delimited segments, assigning each segment to a unique file which will save chronometric information about that one segment apart from data for other segments. Buildfile also "builds" an entire experimental test by chaining together several different passages (in our experiment, the sentences presented were part of a larger whole text, called a "passage"), in the order requested by the experimenter. The result of Buildfile is a unique testfile, to be administered by the computer, which maintains the presentation order of segments within passages, and of passages within the testfile, as selected, delimited, and named by the experimenter.

3. Readit

The testfile created in Buildfile is presented to a subject via the program Readit. Readit displays the testfile segment by segment, passage by passage, to the subject. Between passages are interactive dialogues to cue the subject when one passage is finished; these allow him to control when the next passage starts, and provide a chance for testing or other experimental manipulations between passages. The subject also controls how long and how many times each segment is viewed, pressing a "movement key" when he has finished with one segment and is ready to read the next (or previous) one. In addition to presenting the passages, Readit also stores information about how much time each segment was on the screen, and what order the segments were read in. This data is saved in an output folder named by the experimenter in the form of 3 unique files with the extensions ".trce," ".segs," and ".time." These extensions reflect the reading order of segments, information about each segment, and the amount of time the segment was on the screen each time it was displayed, respectively.

4. Convert

The 3 output files located in the output folder from Readit are combined and manipulated by Convert such that all information from the three files is collapsed in one file. Chronometric and descriptive data are presented for each segment, in the order that the segments were read, for each passage in the testfile. The result of Convert is a unique file, saved into the output folder created by (3) above with the name "time.new."

5. Macwrite/Ms Word

The results of Convert are opened and edited through the "text-only" option in either of these word processors. "0.000" times or extremely large ones resulting from subject mistakes, distractions, etc. are edited out of the timefile in order to clean up the data. The edited file is saved in the given subject's output folder, with a unique name of the experimenter's choosing.

6. Lastone

Lastone calculates summary data from the edited timefile of Convert. The results of Lastone are saved in the output folder in a unique file with a ".conv" extension.

7. Viewdata

BEFORE YOU START

The following manual contains detailed, technical information. Understanding the technical terms used, and the logic behind the examples, will make the manual more palatable. Therefore, the following represents a list of commonly used terms and a short explanation of the examples.

Before the definitions, however, it is crucial that the difference between **file** and **folder** be very clear. Consistent with standard Macintosh lingo, **files** are always subordinate to **folders**. Each time there is "output" from a specific application, it is saved as an output **file** with an extension to tag its origin (see explanation of "extension" below). Every output **file** is saved in an overarching "output" **folder**; hence, when all the applications have been run, a given subject's "output" **folder** contains 5 or 6 different "output" **files**, each with its own origin tag, and each containing the subject's data in a different state of summarization and analysis.

TERMS

Delimiter - a keyboard character which has been chosen to consistently mark the division between two segments in a text. It is simply inserted between the segments of interest, and is later read by the computer as a boundary marker.

Extension - the unique *period + 3-letter* extension added by each application to all files it creates, designed to facilitate identification of output data files. For example, Convert adds ".new" to the file it creates.

Segment - a text subset of any length, created using any standard word processor, and defined by delimiters. Is usually the unit of measurement.

Testfile - the result of **Buildfile** and the input for **Readit**. Contains the input presented to the subject, as s/he will see it.

Text - the group of any characters, including spaces, tabs, numbers, symbols, and letters, which has been created and saved in a word processing program such as **Macwrite** or **Microsoft Word.**

ABOUT THE EXAMPLES

For each step of every application, there are two examples in the manual. One example is written, and the other is a figure. The two examples deliberately reflect different file names to give the reader a wider taste of possible menus, file names, etc. The written example corresponds to the file-naming conventions we used in our experiments; the pictures usually don't.

I. EDITING AND MANIPULATING DOCUMENTS IN MACWRITE or MS WORD

PURPOSE

The purposes for which these two word processors were created are obvious. However, in the following discussion, we will look at some of their lesser-known capacities, with specific interest in the ability to edit and create "text-only" documents. Although slightly different in screen format, **Macwrite** and **MS Word** can be used interchangeably.

CONSTRAINTS

(1) **Macwrite** is by nature slower in every task than **MS Word**. If speed is of high priority, then we recommend using **MS Word**.

(2) **MS Word** has a default pitch which causes the data files resulting from **Convert** and **Lastone** to "wrap". Each time one of these files is opened in **MS Word**, the entire text must be shifted to a lower pitch. This problem does not exist in **Macwrite**, which we recommend using if speed is NOT of high priority.

(3) Only files containing text or data may be opened through a word processor. Programming code, etc. cannot be opened in this manner.

(4) You must be somewhat familiar with the basic functioning of these two word processors; it will be assumed that such things as saving, saving as, closing, quitting, and shade-to-erase operations are familiar to you.

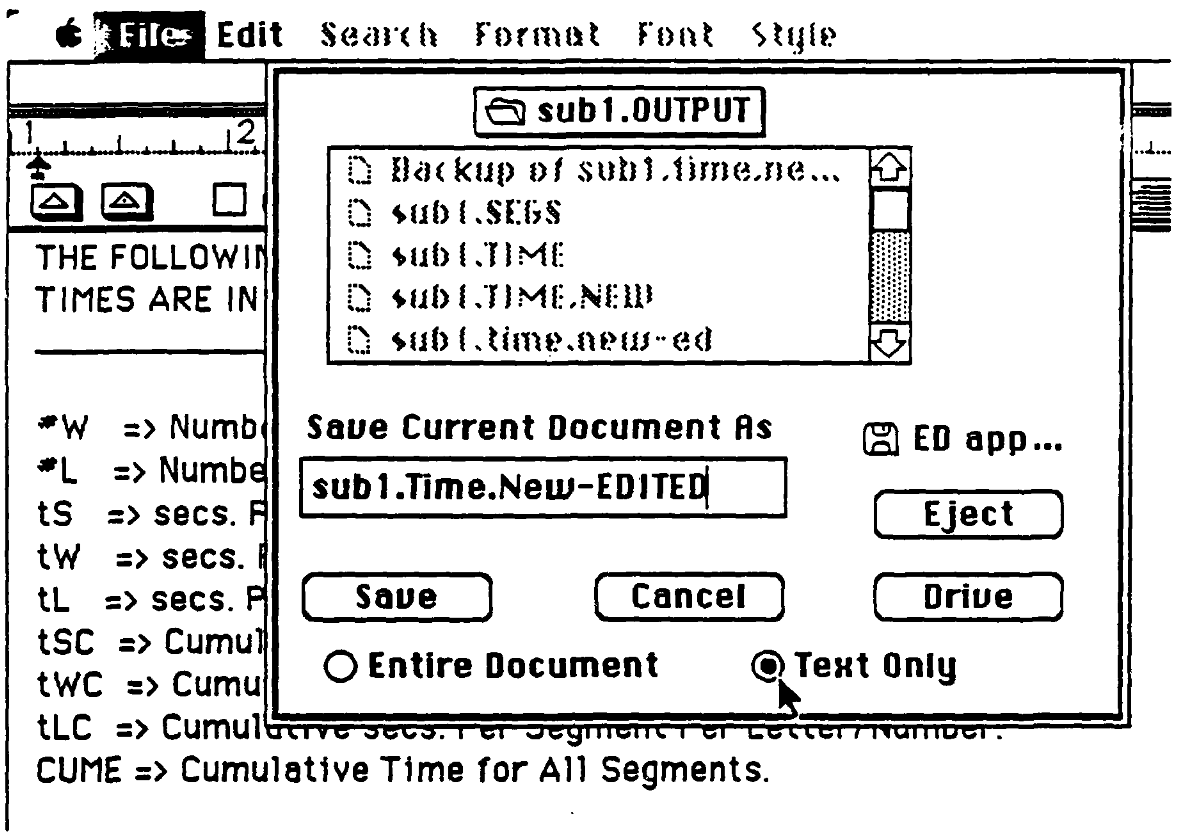
USING MACWRITE or MS WORD

For convenience, we will divide this section into the following two parts: (1) Saving and opening "text-only" files, and (2) Creating and Delimiting the textfiles used to run **Buildfile**.

Saving and Opening "Text-Only" Files

SAVING: After editing a document in either **Macwrite** or **MS Word**, it must be saved. As the process is simpler in **Macwrite**, we will use it for our example. The process is analogous in **MS Word**.

Since **Macwrite** opens all new and all text-only files as "untitled" (see next section), there is no difference between saving your file using the "save" or the "save-as" options. Whichever you choose, you will encounter a dialogue (figure 1) which asks you to provide a name for the as-yet "untitled" file. As you can see, at the bottom of the dialogue are two little circles next to two choices.



Save Current Document As: sub1.Time.New-EDITED

Entire Document [✓ Text Only]

```
THE FOLLOWI
TIMES ARE IN

*W  => Numb
*L  => Numbe
tS  => secs. P
tW  => secs. 
tL  => secs. P
tSC  => Cumul
tWC  => Cumu
tLC  => Cumulative secs. Per Segment Per Letter/Number.
CUME => Cumulative Time for All Segments.
```

-- Figure 1 --

The default choice, which has a filled-in circle, is for regular word-processing documents like this one, and reads "entire document". The other choice, which is the one of interest here, reads "text only".

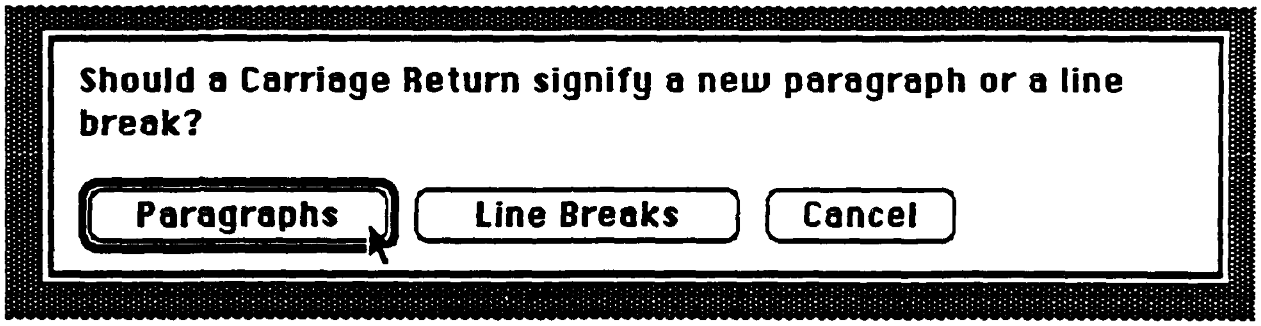
Saving a document as "entire document" means saving all print commands (like pitch, font, style, etc.) and all "page format" commands (such as tabs, indents, rulers, hard returns, etc.). In contrast, saving a document as "text only" means saving only the characters on the screen as they appear (like "gg gg h qp" and the spaces between them), without any other information. This "text only" option corresponds to the "non-document" mode in other word processors, like WordStar.

Before giving your document a name and clicking on the "save" key, you should click on the circle next to "text only". The "text-only" circle will become filled in, and the "entire document" circle will become empty. Thus we see that the two are polar opposites; a file can only be one type or the other. NOW name your document SOMETHING DIFFERENT THAN IT WAS NAMED BEFORE and click on "save". The Mac's saving process is the same as for any other file from here on out. If you try to name the document as it was named before, you will get an error message.

OPENING: To open a text-only file in **Macwrite**, you cannot simply double-click on the text-only file's icon. If you do, you will get the error message which says roughly, "This application is missing and/or in use, and cannot be opened".

Therefore, you must open the file by FIRST opening **Macwrite**. Simply double click on the **Macwrite** icon. Then, once the empty text screen is showing, CLOSE THE SCREEN. Next, drag down on "file" and "open". You see the familiar "open" choices for **Macwrite**. Select the file of interest, and open it. You immediately are presented with a dialogue (figure 2) offering you a choice between "paragraphs" and "line breaks". SELECT PARAGRAPHS. Then you are asked to approve or cancel the file being opened as "untitled". Click on "OK" and your data will appear on the screen.

When you close your text-only file, the same dialogue (figure 2) with the "paragraph" vs. "line-break" choice will appear. Again, SELECT PARAGRAPHS.


Should a Carriage Return signify a new paragraph or a line break?

-- Figure 2 --

Creating and Delimiting textfiles

Creation of textfiles is simple. Using either **Macwrite** or **MS Word**, type the passage, characters, or patterns of interest into the file just as you would any other text. Be advised that double-spacing, boldface, tabs, paragraph indentations, etc. are all superfluous, and might as well not be done; therefore, single-spaced typing is simplest. The only way to achieve any centering, etc. is to space the character(s) involved over by hand. Make sure that you save your finished textfile as "text only" (see previous section).

Delimiting textfiles is also simple. First, select some keyboard character which you are 100% certain DOES NOT appear in **any** of your texts, and designate it as your segment delimiter, i.e. marker. Our delimiter was the black dot created by pressing "option" and the regular keyboard's "8" key at the same time, and it looks like this: •. However, you may choose whatever character suits your fancy. Second, decide where you want the divisions between segments to lie. For us, the division lay at the end of each sentence. Third, put a delimiter between each segment. Using our example, the sentences would look like this.•The black dot would tell **Buildfile** to make a new segment.

You do NOT need a delimiter before the first segment of a passage, nor after the last segment. Also, if a space appears after the delimiter, then there will be a space before the first word of the segment in which it appeared, making it have a small indent when it appears on the screen.

You must keep each individual passage in a separately named textfile

OUTPUT

The result of creating or saving some file as "text-only" will be an icon resembling the typical **Macwrite** icon, except that instead of what appear to be tiny letters on a piece of paper, the "text-only" icon will have just straight lines on a piece of paper.

Any file which has been saved as text-only must be opened through **Viewdata** or **Macwrite/MS Word** as mentioned above.

II. THE APPLICATION BUILDFILE

PURPOSE:

Buildfile was designed to create input for **Readit**. **Buildfile** divides each individual text passage into the number of "segments" that the experimenter desires. In addition. **Buildfile** concatenates several "segmented" passages into an entire test, which is saved on disk and referred to as an "input folder" or "testfile".

First, **Buildfile** operates on specific delimited text (see **Macwrite** manual for more information on delimiting), dividing it into the number of segments marked by delimiters. During this "segmentation" process, **Buildfile** creates a list of all the segments, and their serial order, so that later calculations on reading time and number of words per segment can be made. **Buildfile** can be used to create segments with lengths ranging from one letter up to screen-length paragraphs, and need not be used exclusively for text; numbers or keyboard symbols are also permissible input sources and segments.

Second, **Buildfile** concatenates up to 35 different, individually segmented passages into a complete test. The assumption is that each passage will represent a different type of reading treatment, and that at least 3 passages will be presented to each subject in order to have a balanced design. If your expenment examines, say, individual letter recognition, then the passage distinction may lose significance. In addition, **Buildfile** creates a list of all the passages, and their serial order of presentation, such that **Readit** can effortlessly present the correct passages in the desired order to the subject. **Buildfile** thus maintains two separate levels of design: that of the segment and that of the passage.

CONSTRAINTS:

(1) The texts on which **Buildfile** operates must have been saved as "text only" files. See section on **Macwrite** for more information. "Text only" files are not capable of retaining fonts, pitches, italics, etc.; therefore, issues dealing with unusual letter size, shape, etc. are inappropriate for this application.

(2) The texts on which **Buildfile** operates must contain some sort of delimiters between the desired segments. Again, see the **Macwrite** chapter for more information.

(3) Any segment being submitted to **Buildfile** must have no more than 684 characters (including blanks) - the limit of the **Readit** screen. Any additional requirements will necessitate program adjustments.

(4) Any given text on which **Buildfile** operates may be divided into NO MORE than 30 segments. If more segments are needed, then program alterations will be required.

(5) Although **Buildfile** is capable of creating passages indefinitely, later applications have limits; therefore, NO MORE than 35 passages may be concatenated with **Buildfile**. Again, additional passages will require program alterations.

(6) When naming passages in both **Buildfile** and **Macwrite**, the experimenter must take care to enter NO SPACES between the number and letter characters which comprise a legal passage name. **Buildfile** will not suffer for these errors, but later programs will bomb. For example, "Passage1" and "firstone" are legal; "Passage 1" and "first one" are not.

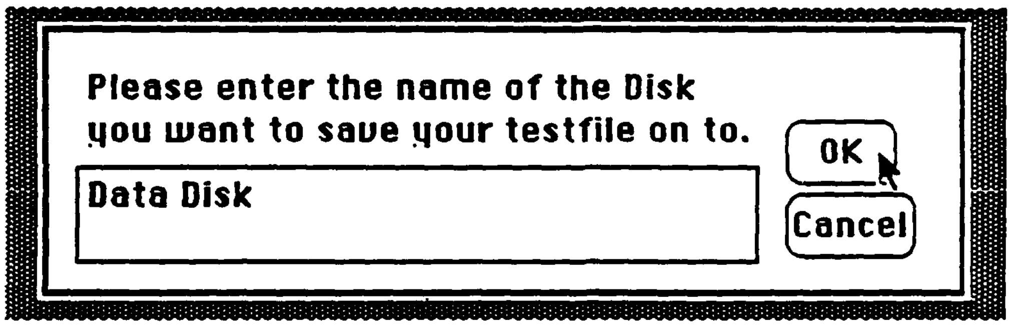
(7) The first passage name entered when **Buildfile** is begun will always be assumed by **Buildfile** to be a sample item, used for training the subject before the real task begins. The time data resulting from it will not appear in the final output from **Readit**. Any additional requirements will necessitate program alterations.

(8) **Buildfile** will NOT automatically counterbalance your design. YOU are responsible for knowing what order you wish your passages to appear in, whether the segments or passages are of comparable length, difficulty, etc., and how many passages you have put into each testfile.

RUNNING BUILDFILE

Before running the application, you must have done four things: (1) Inserted the disk containing the delimited passages on it; (2) Inserted the disk containing **Buildfile** on it; (3) Inserted the disk on which you want the testfiles to be stored; and (4) Decided upon a specified research design, including a naming convention. Requirements (1) - (3) can be located on the same disk, if desired; however, if more than one replication of your entire design will be run, it will be easier to make (3) a disk separate from (1) and (2).

To begin, double click on the icon for **Buildfile**. The following dialogue box (Figure 3) appears, asking you to name the disk onto which you want to save your data ((3) from above). You may elect to use a disk in the hard, internal, or external drives. You must know the name of the disk and type it in exactly as it appears on the icon. When you have typed it in correctly, then click on "OK". If you decide to abort **Buildfile** at this point, click on "cancel".



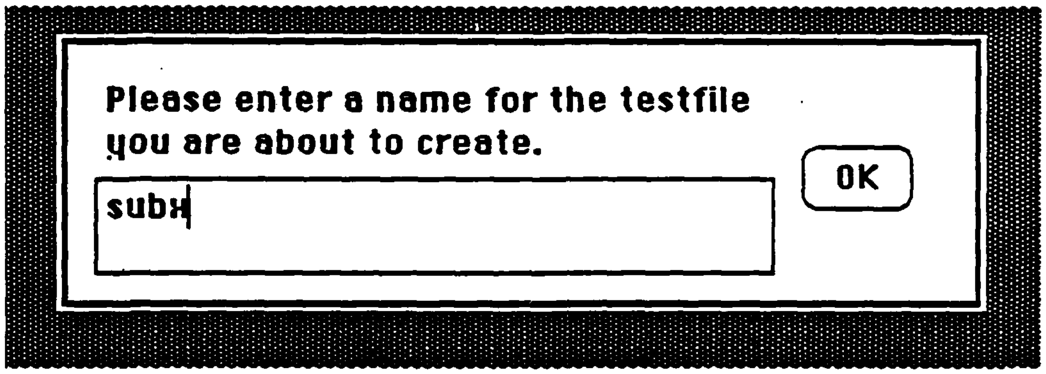
Please enter the name of the Disk you want to save your testfile on to.

Data Disk

-- Figure 3 --

The next dialogue (Figure 4) asks you to assign a name to the input folder into which **Buildfile** will momentarily save all the passages and segments you create. Type in the name, **with no spaces**, and click on "OK" (or hit "return) when you have the name as you desire it.

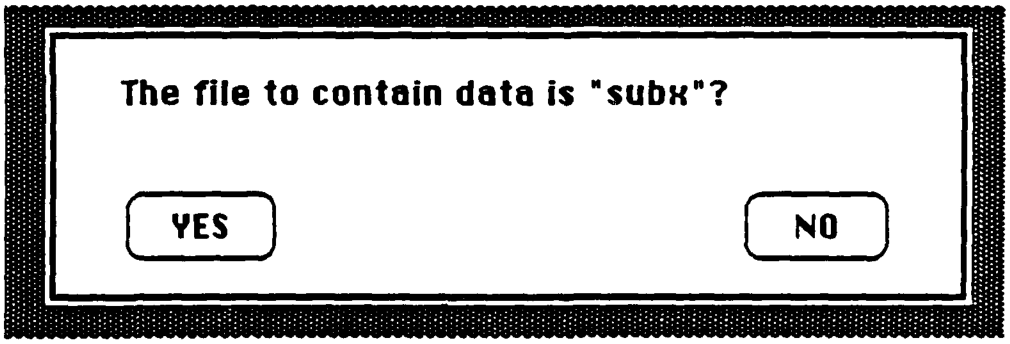
SPECIAL NOTE: **Buildfile** automatically adds to every input folder name a ".input" extension, which makes it easier to identify input files; thus, naming your input file "Sub3" will result in a file titled "Sub3.Input". Do *Not* type the ".input" extension in your file's name - if you do, you will wind up with file names like "sub3.input.input".


Please enter a name for the testfile you are about to create.

subx

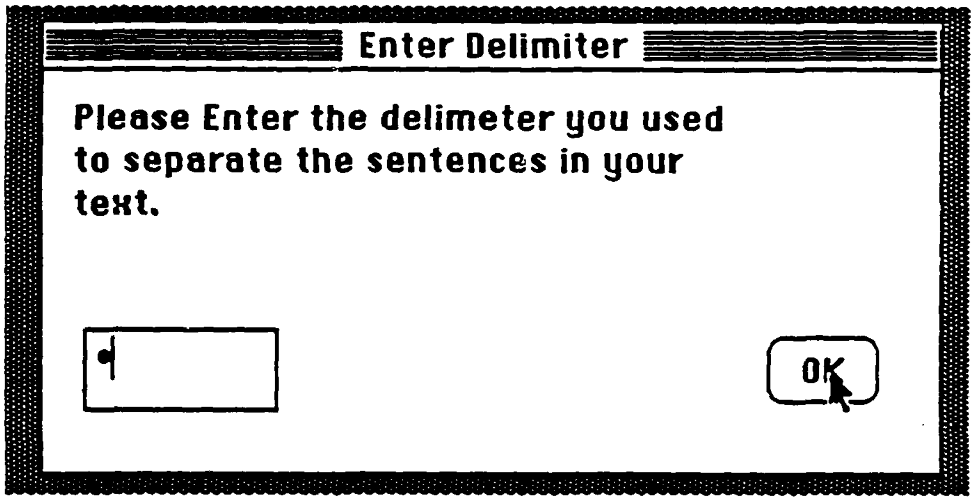
-- Figure 4 --

Once you have finished naming your testfile, a third dialogue (Figure 5) appears, asking for verification of the name and permitting error modification. If the name is incorrect, clicking on "NO" returns you to the second dialogue (Figure 4) for another naming attempt. If it is correct, clicking on "YES" (or hitting return) will move you on to the next stage of the program.


The file to contain data is "subx"?

-- Figure 5 --

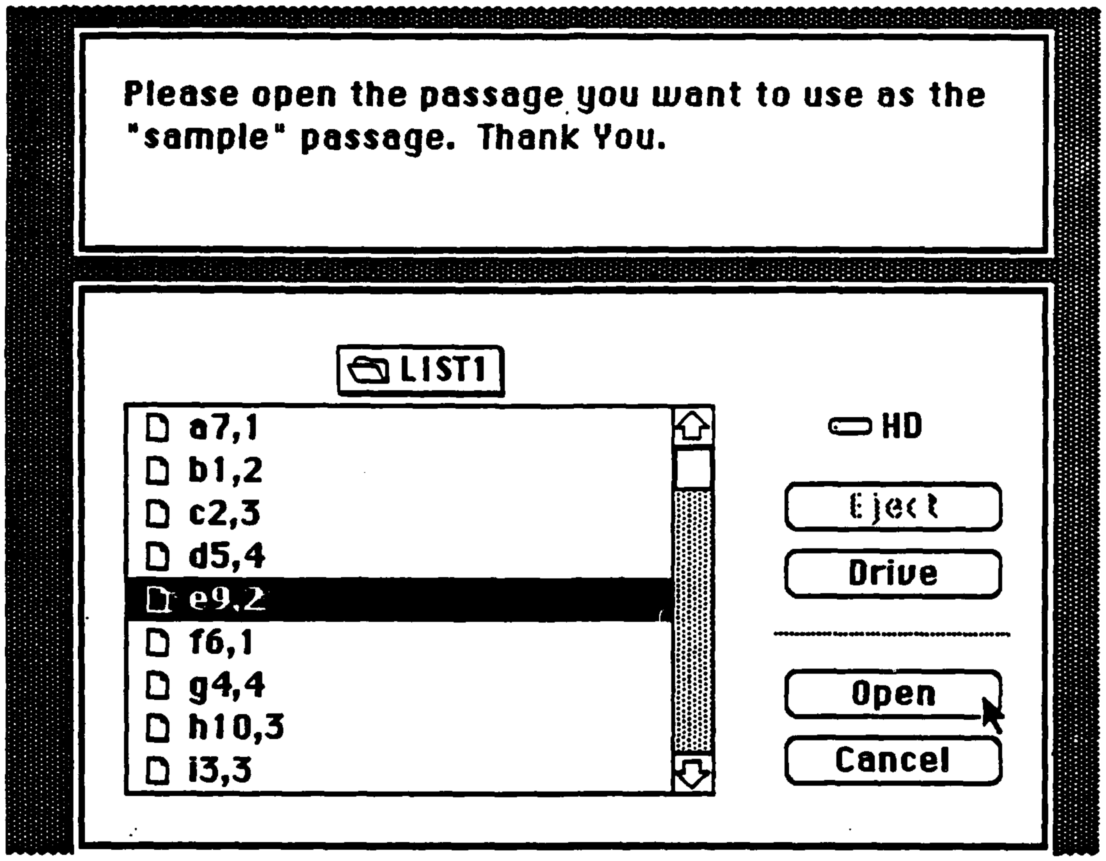
Now **Buildfile** needs to know how to distinguish the segments from each other, so a dialogue box (Figure 6) asks you to type in the DELIMITER you used to differentiate between segments in your text file (see **Macwrite** manual for more information). Once you have typed it in, click on "OK".


Enter Delimiter

Please Enter the delimeter you used to separate the sentences in your text.

-- Figure 6 --

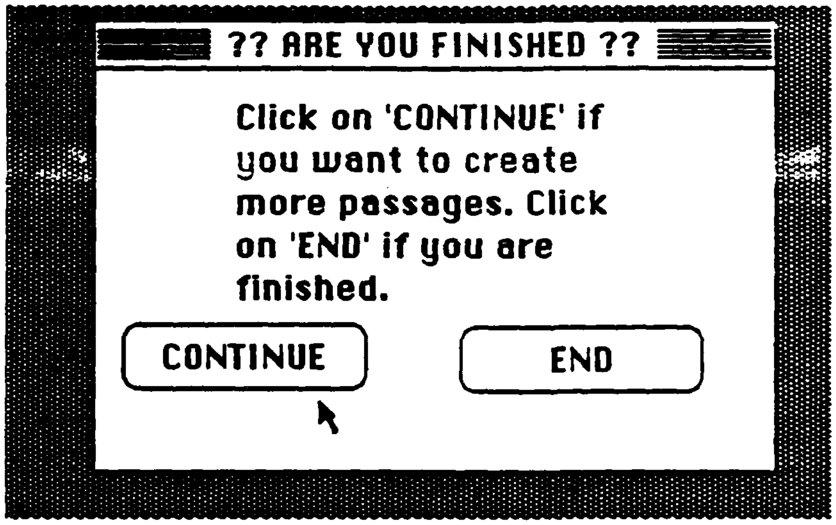
A fifth dialogue appears (Figure 7), requesting that you identify and open the passage you wish to use for your "sample". BE SURE YOU CLICK ON THE SAMPLE AND NOT ON PASSAGE #1. Once you have started the building process, there is no way to reorder the passages you have selected.


Please open the passage you want to use as the "sample" passage. Thank You.

-- Figure7 --

Next, a message appears, requesting you to be patient while **Buildfile** divides up the text into its allotted segments, etc. The message can last anywhere from 10 seconds to 2 minutes, depending upon the length and number of segments in your text.

The final dialogue (Figure 8) offers the choice of either ending **Buildfile** or recursively returning to dialogue #5, which will ask you to select the next text you want to use for passage #1, #2, etc. as appropriate. If you want to add another passage to your testfile, then click on "continue". If you have finished building the testfile, then click on "end". You will be returned to the desktop.



?? ARE YOU FINISHED ??

Click on 'CONTINUE' if you want to create more passages. Click on 'END' if you are finished.

-- Figure 8 --

If you want to build more testfiles, then you must reinvoke **Buildfile** and go through the processes named above, creating another unique output folder.

OUTPUT

The results of **Buildfile**'s operations will be saved in the folder, and on the disk, which you specified at the beginning of the application - that is, you will see a folder with the name you gave it and a ".input" extension. So for example, my testfile will be saved in a folder called "Sub3.input". Inside this folder will be a unique file for EACH SEGMENT of EACH PASSAGE you used to generate your testfile, and one additional file with the same name as the file folder; for example, a file called simply "sub3".

It is strongly recommended that all of **Buildfile**'s results be checked before being run on **Readit**. To check the contents of the testfile you just created, click on the folder icon (for example, "sub3.input"); move to the one entry in the file which is named identically to the file's name (for example, "sub3"); and open it. The contents of this file provide a chronological listing of the passages you entered, and of the segments within them. If they are not in proper order, you should rebuild your testfile. If there are segments missing, you should return to your text files and be certain that all delimiters were properly entered, and that you used the same delimiter in every text. Then you should re-build the file and check it again.

If you now want to run a subject, then you should use the application **Readit**.

III. THE APPLICATION READIT

PURPOSE

Readit presents to subjects the contents of a testfile created by **Buildfile**. These contents are presented one segment at a time, passage by passage, on a computer screen. **Readit** records both processing time per segment, and the reading order of segments, into an output file.

Subjects control the speed of segment presentation by pressing a key to finish display of the current segment and begin display of the next segment. Processing time is defined as the interval between a given segment appearing on the screen, and the press of a movement-key; it is automatically scored by the computer to within 1/60 of a second, and is recorded by **Readit** in the output folder, in a file with a ".time" extension.

Subjects control the order of segment presentation; once they have read at least two segments in a given passage, they can either call the next segment OR recall the previous segment to the screen. Therefore, when they have finally finished "reading" a passage to their satisfaction, the subjects may have many reversals in their reading order. For example: they may have read segments 1,2,3,4,5,6; then returned to 5; then to 4; then read 5,6,7; then returned to 6; then read 7,8,9; then returned to 8; then to 7; and so on. **Readit** keeps track of the sequence of segments as they are read by the subject, and records the order into the output folder, in a file with a ".trce" extension.

The subjects may also have read a given segment any number of times. From the example above, segments 4 and 8 were read twice each; segments 5,6, and 7 were read three times each. **Readit** keeps track of the number of times each segment was read per passage, and stores that information into the output folder, in a file with a ".segs" extension.

Subjects cannot return to segments from a previous passage once they have finished with that passage and begun a new one; thus, the integrity of given passage treatments is maintained.

CONSTRAINTS

(1) The testfile used as input for **Readit** must have been created by **Buildfile** and must have NO SPACES in either its name or in the names of passages within it.

(2) The keyboard used when running **Readit** should be the modern, expanded keyboard with a numeric keypad. If the older keyboards are all that are available, program alterations will be required.

(3) Although **Readit** results in time data to three decimal places, time measurements are really only accurate to 1/60th of a second (.016), which corresponds to one

Macintosh "tick". The 3rd decimal place is automatically rounded. Time which elapses between passages is not recorded.

(4) While reading, subjects are not able to scroll forward by holding down the "forward" key; that key must be pressed once for each segment. However, subjects may scroll backward by holding down the "previous" key.

(5) Subjects may re-read any segment which has been read at least once. They may not "finish" reading a passage, i.e. stop the timer, until each segment has been read at least once. If they accidentally "finish" before having re-read specific segments to their satisfaction, there are provisions for reentering the passage. However, if they accidentally "begin" a passage before they are truly ready to read, there is no way to erase the "false start."

RUNNING READIT

Before beginning **ReadIt**, you must have done the following things: (1) Inserted the disk containing the **ReadIt** application; (2) Inserted the disk containing the input folder, or testfile; (3) Inserted the disk onto which you wish to save your subject's output file; (4) Made certain that the keys corresponding to "next," "previous," and "finished" are clearly labeled for the subject's reference; and (5) Explained to the subject how to use the keys - the "sample" passage is usually when this is done, but it is important even before starting **ReadIt** that the subject understand the extent to which s/he has control over the segment viewing process.

The three movement keys are the following:

"Previous" is the "Apple" (Control) key
"Next" is the "0" key on the numeric keypad
"Finished" is the "Enter" key on the numeric keypad.

Refer the Figure 9 for location on the keyboard. The shaded keys are those of interest.

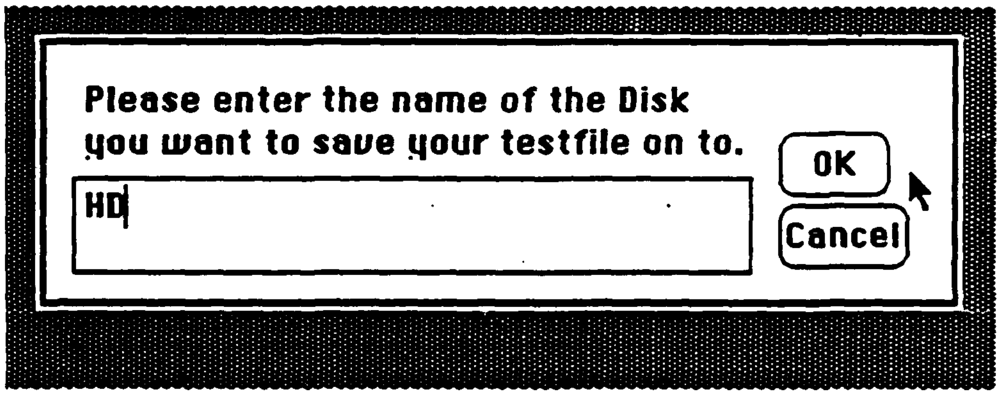
LOCATION OF PREV, NEXT, AND FINISH KEYS
ON THE TYPEWRITER KEYBOARD

-- Figure 9 --

In case of confusion, the bottom of every screen explains what the movement keys are and what they do, and is updated with each new screen. For example, the option to "finish" only appears when the last segment of the passage is displayed on the screen for the first time. "Finish" remains as an option from then until the subject has finisnec reading that particular passage.

If the subject attempts to press a key which is either illegal (for example, the "return" key) or which is not valid at that point (for example, the "finished" key before having read every segment at least once), the machine "beeps" to alert subject and experimenter to the error.

To begin **Readit**, double click on the **Readit** icon. The following dialogue (Figure 10) appears, requesting that you specify which disk you want the output data to be saved on. Type in the disk name exactly as it appears on the disk's icon, and click on "OK". You may elect to save your data on a disk in the hard, internal, or external drives; it may be saved on the same disk as that containing the input files, or on its own unique disk. If you started **Readit** accidentally, click on "cancel" to return to the desktop.

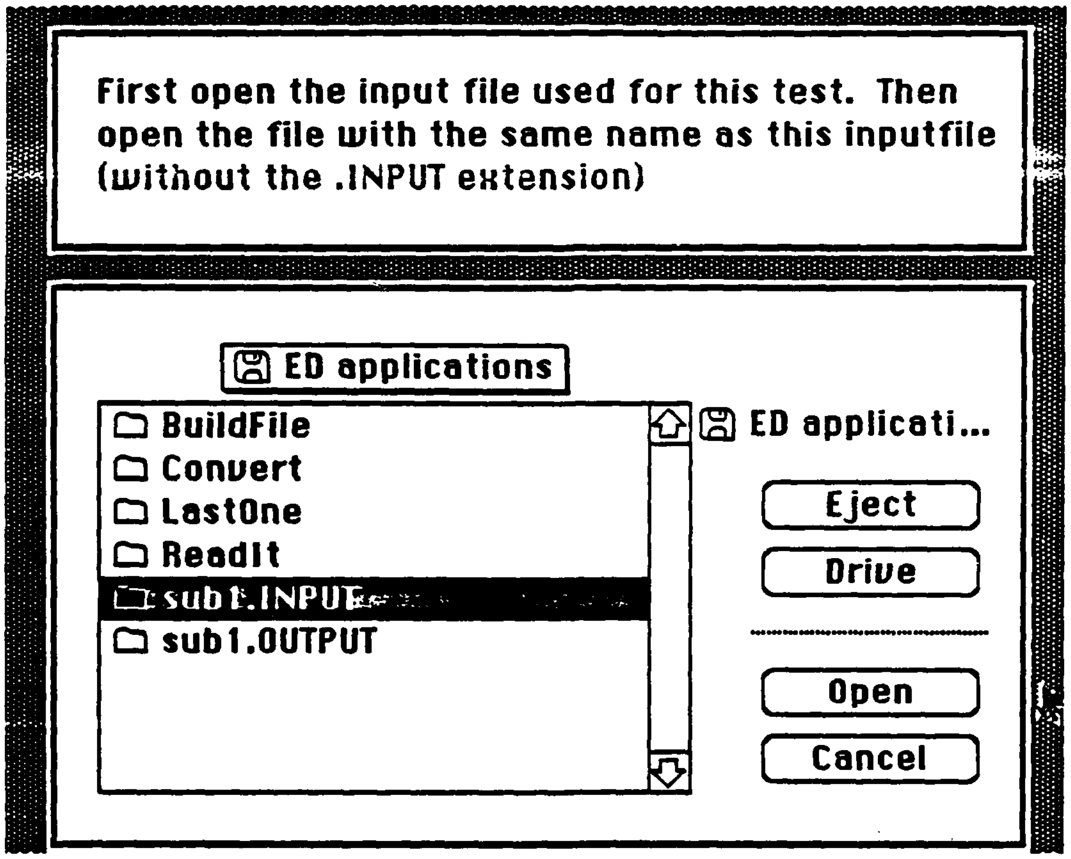

Please enter the name of the Disk
you want to save your testfile on to.

HD

-- Figure 10 --

The next dialogue (Figure 11) is really a 2-part dialogue, and requires a 2-part action.

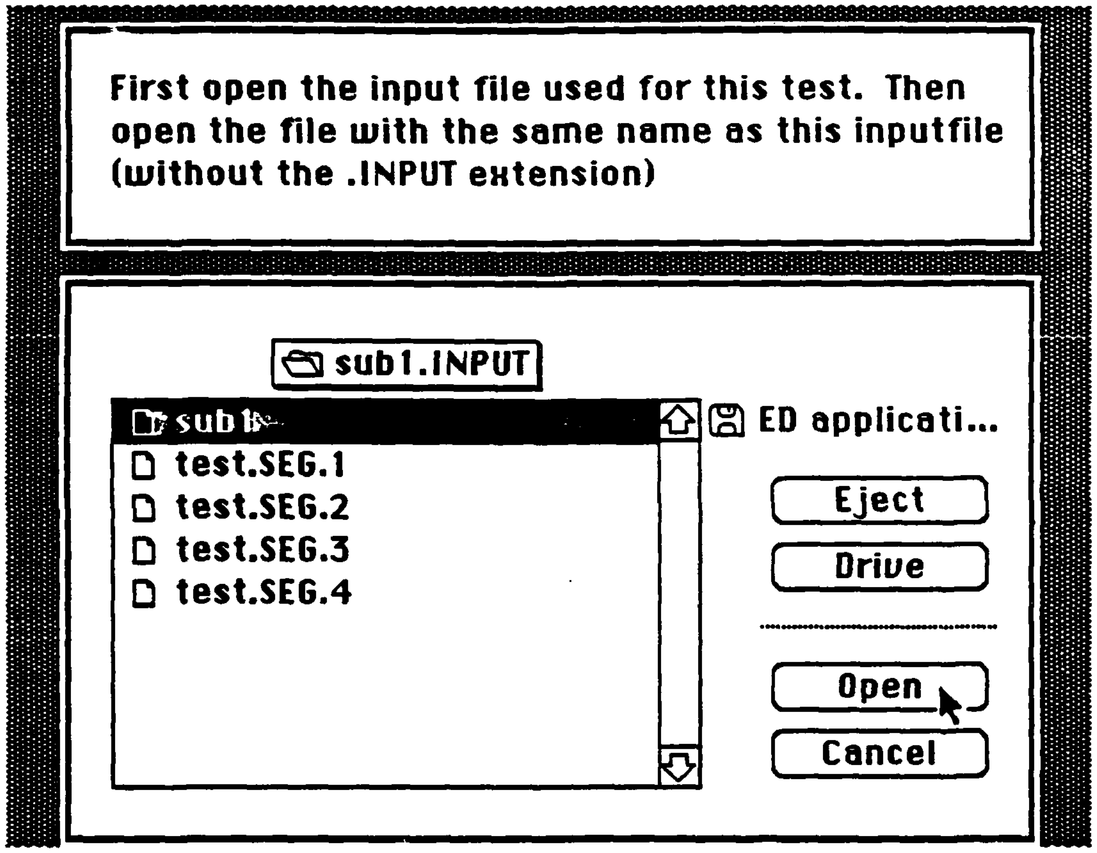
First, follow the instructions in the first step of the dialogue (Figure 11) and open the input file you created with **Buildfile**. The dialogue header will stay the same, but the listing will change from all folders on that disk to the contents of the folder you have selected.



First open the input file used for this test. Then open the file with the same name as this inputfile (without the .INPUT extension)

-- Figure 11 --

Second, follow the instructions in the second part of the dialogue, and open the file with the same name as your input folder (Figure 12). For example, let us assume that your input folder is called "Sub3.Input". You have opened this folder and now you scroll through its contents until you come across the file called simply "Sub3" (Figure 12). Open this file.



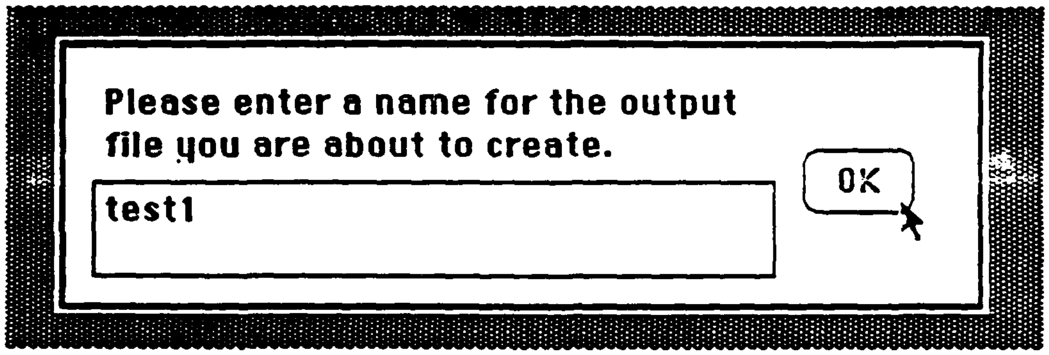
First open the input file used for this test. Then open the file with the same name as this inputfile (without the .INPUT extension)

-- Figure 12 --

(Opening the input folder tells **ReadIt** exactly which folder the input lies in; and more specifically, which file the *list* of order for the input passages is located in.)

Now **ReadIt** asks for an output file name (Figure 13). (This "file" will actually become a folder, so we will refer to it as a folder in this text.) For simplicity's sake, we recommend titling it with the same root name that your input folder had. So for example, if the input folder were named "sub3" then the output folder name should be "sub3". **ReadIt** will generate a ".output" extension for the output folder; therefore the output folder will be titled "sub3.output".

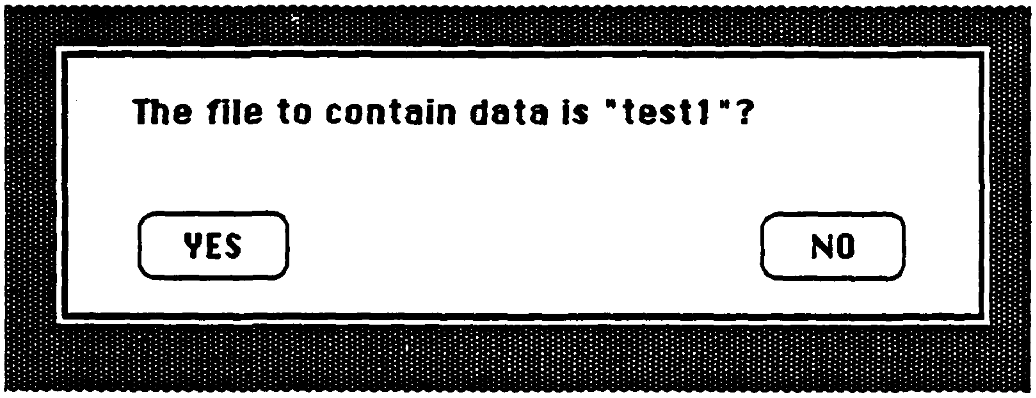
IMPORTANT NOTE: similar to **Buildfile**, the user should *NOT* add the ".output" extension - **ReadIt** will do this for you. Simply type in the desired root name, for example "Sub3". Once you have entered the name, click on "OK" to continue forward in the program.


Please enter a name for the output file you are about to create.

test1

-- Figure 13 --

The next dialogue (Figure 14) offers the experimenter a chance to modify the output folder name in case of typos or other errors. Click on "yes" if the folder name is correct; click on "no" if it is incorrect, and **Readit** will return you to the dialogue in Figure 13 for another chance.

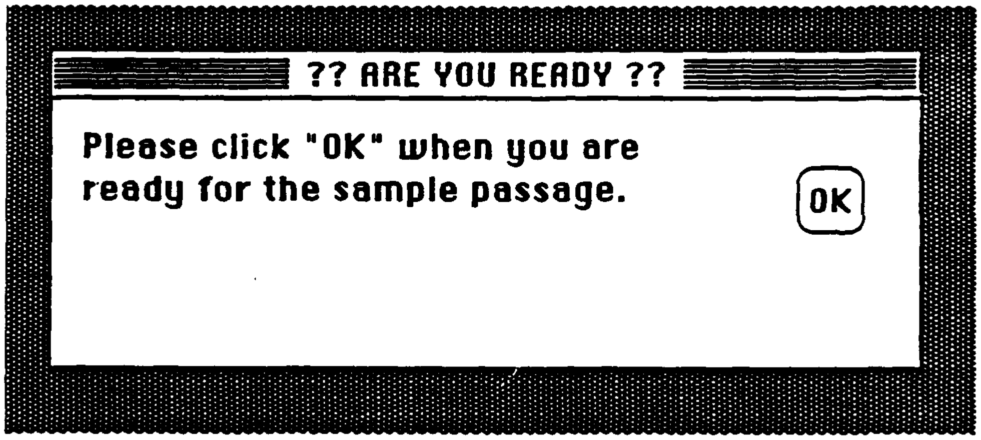

The file to contain data is "test1"?

-- Figure 14 --

Readit has only 2 more dialogues, and they are for the subject's use only; therefore, it is imperative that both experimenter and subject be completely familiar with these dialogues.

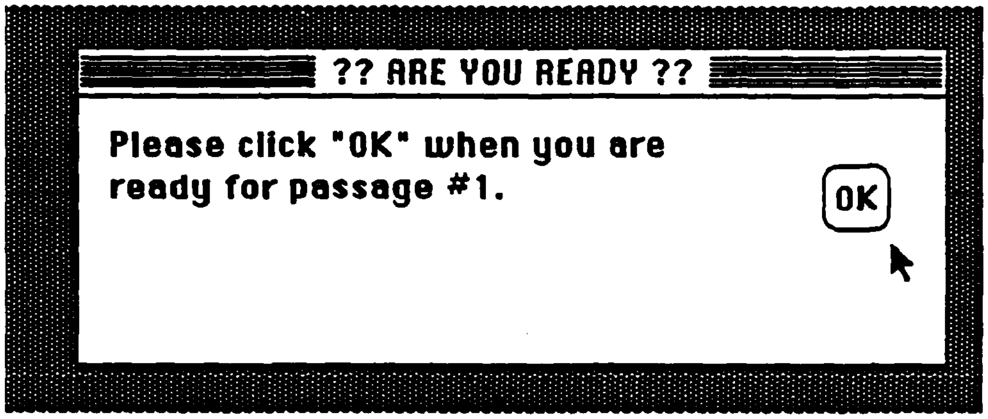
The first type of dialogue precedes each new passage (Figures 15 and 16), and is in a sense the start/stop button on the Macintosh's internal stopwatch. The internal timer does not begin until the "OK" button is clicked; it is activated once the "OK" button is clicked on and the first segment is on the screen. *It is extremely important that the subject does* ***not*** *click on OK until s/he is actually ready to begin reading the passage.*

Each time the "Are you ready?" dialogue (Figures 15 and 16) comes up on the screen, it mentions the sequential number of the passage to be displayed; the only exception is that the "sample" passage has no number. Thus, in Figure 15 the sample passage will begin once "OK" is clicked; in Figure 16, the first passage will be presented.


?? ARE YOU READY ??

Please click "OK" when you are ready for the sample passage.

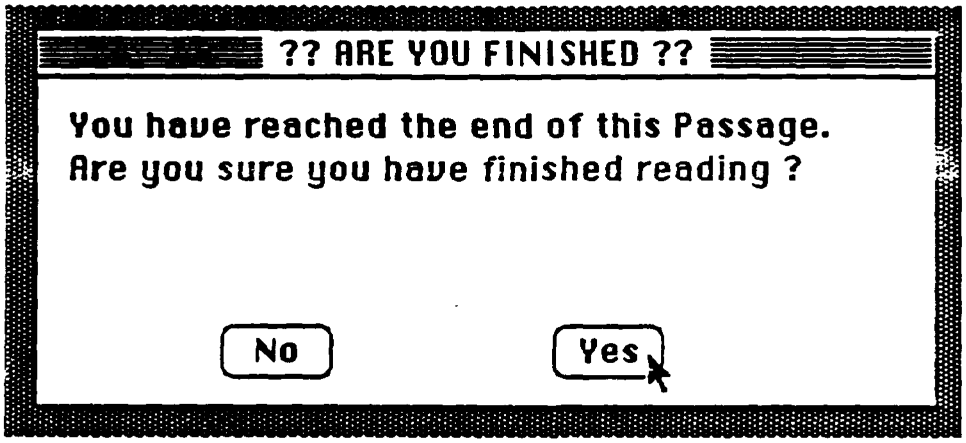
-- Figure 15 --


?? ARE YOU READY ??

Please click "OK" when you are ready for passage #1.

-- Figure 16 --

The second type of dialogue (Figure 17) follows each passage and is the "stop" button of the internal stopwatch. It occurs after each passage has been read and the subject has pressed the "finished" key to exit the passage. This dialogue gives the subject a choice: either quit this passage or reenter it. Clicking on the "no" key puts the last segment of the passage just "finished" back onto the screen, with normal scrolling options open, and the timer continuing to be active. Clicking on the "yes" key finishes the session with that passage and STOPS the timer. The given passage can never be returned to by any single subject once the "yes" key has been clicked on.


?? ARE YOU FINISHED ??

You have reached the end of this Passage.
Are you sure you have finished reading ?

-- Figure 17 --

It is very important that the subject respond to this dialogue because the program cannot write the time data into its location until "yes" has been clicked on.

Finally, once the last passage has been read and "yes" has been clicked on, there is a small notice which appears to inform you that the machine is writing all the time, trace, and segment information into the appropriate output file. Once the writing process is completed, the desktop appears and any other application may be selected.

OUTPUT

The output from running **ReadIt** will be saved into the output folder and on the disk you specified at the beginning of the test session. It is highly recommended that you double-check the contents of the output folder generated from each session of **ReadIt** in order to be certain that the program worked properly.

Inside the output folder should be 3 files, each with your original root name and a unique extension. For the sake of simplicity, we will continue the example from above Therefore, the three files in our output folder should be: "sub3.time", "sub3.segs", and "sub3.trce". Remember, the ".time" file contains the record of each segment's reading time; the ".segs" file contains information about each segment; and the ".trce" files provide additional information about segment reading order and the number of letters, numbers, words, etc. in each segment.

If these three files are missing, check other disks, other folders with similar names that you could have accidentally typed, etc.

These three small files will be merged when you run them through the **Convert** application; therefore, never throw away the contents of your output file just to make room on the disk.

Although it won't be very useful to you at this point, the data in these three files can be printed. Simply use **Viewdata** to print (see manual for **Viewdata**).

IV. THE APPLICATION CONVERT

PURPOSE

Convert unifies the time, reading order, and segment data collected by the **Readit** application and performs additional calculations upon that data. These calculations will be discussed in detail in the "output" section of this manual.

CONSTRAINTS

(1) All three of the subfiles produced by **Readit** must be in the output folder for a given subject, and must have data in them.

(2) **Convert** will automatically save its results by placing them into the output folder containing the unconverted results from **Readit**. We recommend keeping all data pertaining to the same subject together in one output folder. Therefore, you must be certain that the disk containing the output folder of interest has enough space on it for the results of **Convert** (roughly 5-50k).

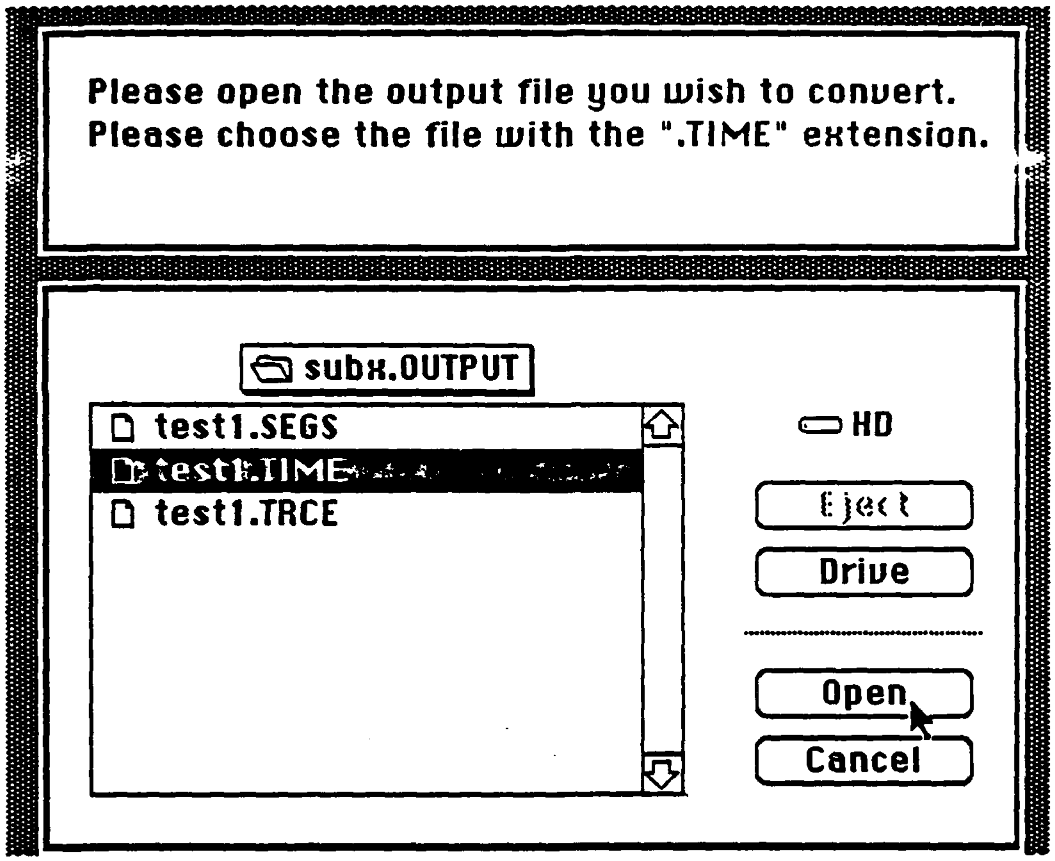
RUNNING CONVERT

Before starting, you must have done the following: (1) Inserted the disk containing the input folder(s) used for running **Readit;** (2) Inserted the disk containing the output folder(s) resulting from running **Readit**; and (3) ascertained that there is enough space on the disk containing the output file for you to add about 5k of "converted" timefiles.

To start **Convert**, double click on its icon. The first dialogue to appear (Figure 18) is really a two-part dialogue requiring a two-part action. It asks for the folder and file name of the data to be considered.

First, select and open the folder containing data you wish to convert. So, continuing the example we used in the **Buildfile** and **Readit** manuals, I would select the folder named "sub3.output".

Second, once within the output folder, select the file with the ".time" extension. To continue my example, I would select "sub3.time" and open it.

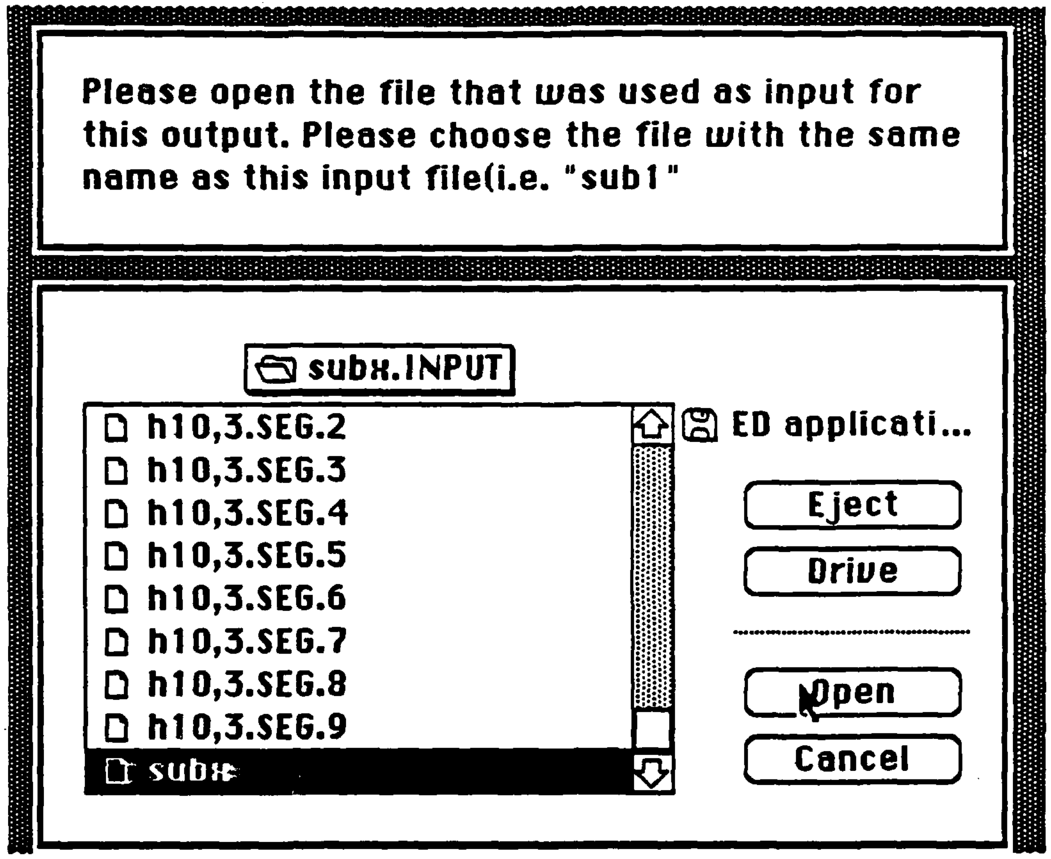

Please open the output file you wish to convert.
Please choose the file with the ".TIME" extension.

-- Figure 18 --

The second dialogue to appear (Figure 19) is also a two-part dialogue requiring two actions. It asks for the folder and file name of the input from which this data was generated, so that it can match, segment-for-segment, character-for-character, the data to the input.

First, select the input folder you used to get the data from **Readit**. So, continuing my example, I would go to the disk containing "sub3.input" and open that.

Second, select (from the contents of the input folder) the file with the same root name as the input file, i.e. "sub3", and open it.

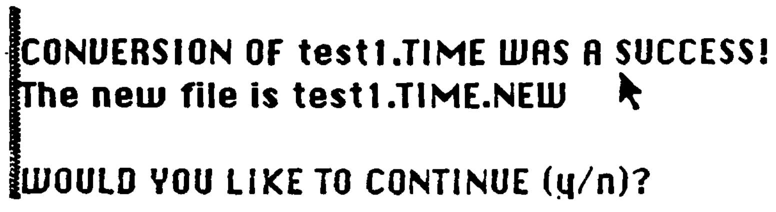

Please open the file that was used as input for this output. Please choose the file with the same name as this input file(i.e. "sub1"

-- Figure 19 --

Now **Convert** will perform calculations and combinations on your data, and the screen will be blank. The time necessary to **Convert** data ranges from 1 second to 3 minutes, depending upon the number of segments, passages, etc. in your testfile, and the number of times they were viewed by the subject.

The final **Convert** dialogue (Figure 20) will appear at the bottom of an otherwise blank screen and will inform you that "the conversion was a success!" (If it wasn't a success, see below.) It will also tell you what the converted file's name is. Again, **Convert** has been written to automatically add a ".new" extension to the data it converts. Therefore, if you were to submit "sub3.time" to **Convert**, the resulting new file would be named "sub3.time.new".

This last dialogue (Figure 20) also asks if you would like to perform another conversion on a separate file. If so, type in "y" and hit return; you will be returned to **Convert's** dialogue #1. If you do not wish to perform another conversion, then type in "n" and hit return; you will get a message which says "normal program termination. Hit enter to return to shell". This is computerese for "hit return to get back to the desktop".



```
CONVERSION OF test1.TIME WAS A SUCCESS!
The new file is test1.TIME.NEW

WOULD YOU LIKE TO CONTINUE (y/n)?
```

-- Figure 20 --

If you receive an error message or a bomb, you should check the data in your output folder using **Viewdata** (see manual). You might also check for a disk error in either the input or the output folders. Finally, you may have inadvertantly exceeded the 35 passage limit stated in the beginning of **Buildfile**'s manual; **Convert** will not process more than 35 passages.

OUTPUT

Table 1 presents a sample of actual data resulting from **Convert.** The top section defines the headers used for the data; the bottom section presents data from the first passage read by the subject. (For the sake of brevity, we did not include data from all 8 passages presented to our subject; however, the 7 remaining passages would all look similar to the one shown here.)

In this example, the subject's number (7) was the basis for the output folder name: "sub7.output". The first passage that subject 7 read was named "14,1" and there were 19 segments in this passage (we only show 8 of them here). As noted before,1 segment = 1 sentence.

```
************************************************************

THE FOLLOWING OUTPUT IS FOR SUBJECT "sub7."
TIMES ARE IN VIEWING ORDER...

__________________________________________

#W  => Number of Words.
#L  => Number of Letters/Numbers.
tS  => secs. Per Segment.
tW  => secs. Per Segment Per Word.
tL  => secs. Per Segment Per Letter/Number.
tSC  => Cumulative secs. Per Segment.
tWC  => Cumulative secs. Per Segment Per Word.
tLC  => Cumulative secs. Per Segment Per Letter/Number.
CUME => Cumulative Time for All Segments.

*=====================================================*

SEGMENT    #W  #L    tS      tW     tL     tSC     tWC    tLC    CUME
14,1.SEG.1 24 132  22.900  0.954  0.173  22.900  0.954  0.173  22.900
14,1.SEG.2 13  66   5.267  0.405  0.080   5.267  0.405  0.080  28.167
14,1.SEG.3 18  93  31.483  1.749  0.339  31.483  1.749  0.339  59.650
14,1.SEG.4  9  53   6.183  0.687  0.117   6.183  0.687  0.117  65.833
14,1.SEG.5 10  60   0.950  0.095  0.016   0.950  0.095  0.016  66.783
14,1.SEG.4  9  53   0.250  0.028  0.005   6.433  0.715  0.121  67.033
14,1.SEG.3 18  93   5.983  0.332  0.064  37.467  2.081  0.403  73.017
14,1.SEG.4  9  53   6.133  0.681  0.116  12.567  1.396  0.237  79.150
14,1.SEG.5 10  60   9.483  0.948  0.158  10.433  1.043  0.174  88.633
14,1.SEG.6 14  73   8.467  0.605  0.116   8.467  0.605  0.116  97.100
14,1.SEG.5 10  60  17.233  1.723  0.287  27.667  2.767  0.461 114.333
14,1.SEG.6 14  73  10.450  0.746  0.143  18.917  1.351  0.259 124.783
14,1.SEG.7  8  64  10.100  1.263  0.158  10.100  1.263  0.158 134.883
14,1.SEG.8  5  33   2.467  0.493  0.075   2.467  0.493  0.075 137.350
**********************************************************
```

-- Table 1 --

ABBREVIATIONS IN TABLE 1

A LETTER/NUMBER is defined as each character (excluding spaces) in a given segment. Therefore "a" is 1 letter/number; so are "-" and "%". "19" is 2 letters/numbers.

A WORD is defined as a cluster of LETTERS; WORDS are separated by spaces. Therefore "sprig" is 1 word; "sp rig" is two words.

A SEGMENT is defined as the letters and/or words which fall between 2 delimiters in the **Macwrite** textfiles. Therefore, assuming the black ball is a delimiter, both "•The boy•" and "• 2 + 2 = ___•" are SEGMENTS. As we have seen in the **Readit** and **Buildfile** manuals, 1 SEGMENT at a time is presented on the screen, and the subject determines when the next segment will appear.

In the header for the data in table 1, the first item is the passage and segment identifier; items 2-4 are relatively self-explanatory; and items 5-6 and 7-10 represent measures derived from the information contained in items 2-4. Let us take a closer look at these items, using the data in the last row of the printout for our example.

All times are in seconds.

Items 1-4:

"Segment" identifies which passage and segment the data refers to, i.e. "14,1.seg.8"
"#W" identifies how many words are in the given segment, i.e. "5".
"#L" identifies how many letters/numbers are in the given segment, i.e. "33".
"tS" records how much time the subject spent viewing the given segment on this particular view, i.e. "2.467".

Items 5-6:

"tW" uses the information in "tS" and "#W" above to calculate reading time per word within each individual segment, i.e. "0.493".
"tL" uses the information in "tS" and "#L" above to calculate reading time per letter within each individual segment, i.e. "0.075".

Items 7-10:

"tSC" uses the information in "tS" above to calculate the cumulative reading time for an individual segment, i.e. "2.467".
"tWC" uses the information in "tW" above to calculate the cumulative reading time per word for an individual segment, i.e. "0.493".
"tLC" uses the information in "tL" above to calculate the cumulative reading time per word for an individual segment, i.e. "0.075".
"CUME" simply adds together all the times in the "tS' column to provide a reading time total for the passage as a whole, i.e. "137.350".

It should be noted that the information in items 7-9 is calculated for each segment as it was read. Thus, it gives play-by-play information about reading rate for each segment.

SPECIAL NOTE: Depending on the size of segments and the measures of interest in your experiment, you may wish to set parameters for psychologically reasonable reading time, and edit out of the data file any times (i.e., individual lines of data) which fall below the parameter. Our criterion was .7 sec; it was felt that any time less than this was not spent processing the segment on the screen, but rather resulted from key-pressing time during an attempt to skip over the segment.

Naturally, if you edit out several reading times per passage of, say, .5 to .6 secs, the columns containing information about "cumulative reading time to date" and "total reading time for the passage" will no longer be completely correct - they will still reflec the initial, unedited data. This discrepancy will be remedied by using the applicatio **Lastone.** See that manual for more detailed information.

Reading Strategy Information in Table 1

The data resulting from **Convert** not only contain time information. Inherent in its structure is information about when and to which segments a given subject returned t re-read. This information is contained in the sequential order of the segments viewed, listed and read from top to bottom of the timefile.

Looking at the data in Table 1, for example, we see that subject 7 read the first 5 segments in order. Then s/he began to "read backwards", by returning first to segmen 4, and then to segment 3. At this point, the subject recommenced "forward" reading by reading segments 4,5,and 6. Again, however, the subject read backwards by going to segment 5. The remaining data shows forward reading from segments 6 to 8.

As mentioned in the section preceding this one, we might wish to edit out the segment information associated with times less than, say, 1.5 secs. If we were to do so, the data from table 1 would have the following boldfaced lines edited out of it (Table 2):

SEGMENT	#W	#L	tS	tW	tL	tSC	tWC	tLC	CUME
14,1.SEG.1	24	132	22.900	0.954	0.173	22.900	0.954	0.173	22.900
14,1.SEG.2	13	66	5.267	0.405	0.080	5.267	0.405	0.080	28.167
14,1.SEG.3	18	93	31.483	1.749	0.339	31.483	1.749	0.339	59.650
14,1.SEG.4	9	53	6.183	0.687	0.117	6.183	0.687	0.117	65.833
14,1.SEG.5	**10**	**60**	**0.950**	**0.095**	**0.016**	**0.950**	**0.095**	**0.016**	**66.783**
14,1.SEG.4	**9**	**53**	**0.250**	**0.028**	**0.005**	**6.433**	**0.715**	**0.121**	**67.033**
14,1.SEG.3	18	93	5.983	0.332	0.064	37.467	2.081	0.403	73.017
14,1.SEG.4	9	53	6.133	0.681	0.116	12.567	1.396	0.237	79.150
14,1.SEG.5	10	60	9.483	0.948	0.158	10.433	1.043	0.174	88.633
14,1.SEG.6	14	73	8.467	0.605	0.116	8.467	0.605	0.116	97.100
14,1.SEG.5	10	60	17.233	1.723	0.287	27.667	2.767	0.461	114.333
14,1.SEG.6	14	73	10.450	0.746	0.143	18.917	1.351	0.259	124.783
14,1.SEG.7	8	64	10.100	1.263	0.158	10.100	1.263	0.158	134.883
14,1.SEG.8	5	33	2.467	0.493	0.075	2.467	0.493	0.075	137.350

– Table 2 –

Note that if these 2 data lines are edited out of the timefile, the number of times segments 5 and 4 were read will decrease by 1 each. This change will NOT be reflected in the converted timefile; as mentioned previously, it will be taken into account when the application **Lastone** is run on the edited version of this timefile.

It is possible that subjects will have viewed many sequential segments at times lower than your criterion. When you edit these times out, you will have a large gap in the segment numbers. For example, the subject may read forward through segments 1-15; then backward through 14, 13, and 12; and then (after a gap resulting from your editing) through 3,2, and 1. This "gap," and more specifically, the sentences read on either end of it, can provide information about segments that were actually processed (ostensibly important to the subject in some way) versus those which were skimmed over (ostensibly unimportant to the subject).

Taken together, the data contained in the 10 columns of, and the order of the rows crossing those columns in, the **Convert** printout represents a large amount of information about the reading rate, strategy, etc. used by subjects for each individual passage, and across passages.

It is highly recommended that this data be either viewed (using **Viewdata**) or edited (through one of the word-processing packages - see the manual on **Macwrite/MS Word**) in order to be certain that all of the passages which were read appear, in proper order, in the ".time.new" file. If they are not correct, check for spaces in input filenames, missing segments in the file opened in the second part of dialogue #2, etc.

To print the data from **Convert**, simply use **Viewdata** (refer to the manual) and print it out.

V. THE APPLICATION LASTONE

PURPOSE

Lastone performs specific calculations on the (edited) timefile resulting from **Convert.** **Lastone** corrects certain information which was made defunct by editing converted timefile (see **Convert** manual), and creates some new data (such as reading rate per word and total processing time per word, calculated over both segments and passages). A detailed discussion and examples of these calculations is contained in the OUTPUT section of this manual.

In addition to performing specific calculations, **Lastone** gives summary data, such as total reading time per passage and the number of words per passage. (For other applications, one "word" can be conceived of as a cluster of characters without spaces between them; in other words, a space signals a new word.) If other types of summary data are desired, it should not be difficult to modify the **Lastone** program to meet specific needs.

CONSTRAINTS

(1) The timefile upon which **Lastone** performs can only be the result of **Readit and Convert.** Other program results without the same specific format are not useable.

(2) The timefile you wish to submit should have been EDITED to meet your plausibility and subject error constraints (see **Macwrite** manual for the editing information). Calculations based on the uncorrected timefiles will be fraught with abnormally high or low numbers.

(3) There must be enough room on the disk where your output file lies to add another small file; **Lastone**'s output will join the ".time", ".time.new", and ".time.new-ed" files in your subject's output file. See Table 3 for a sample of **Lastone**'s output.

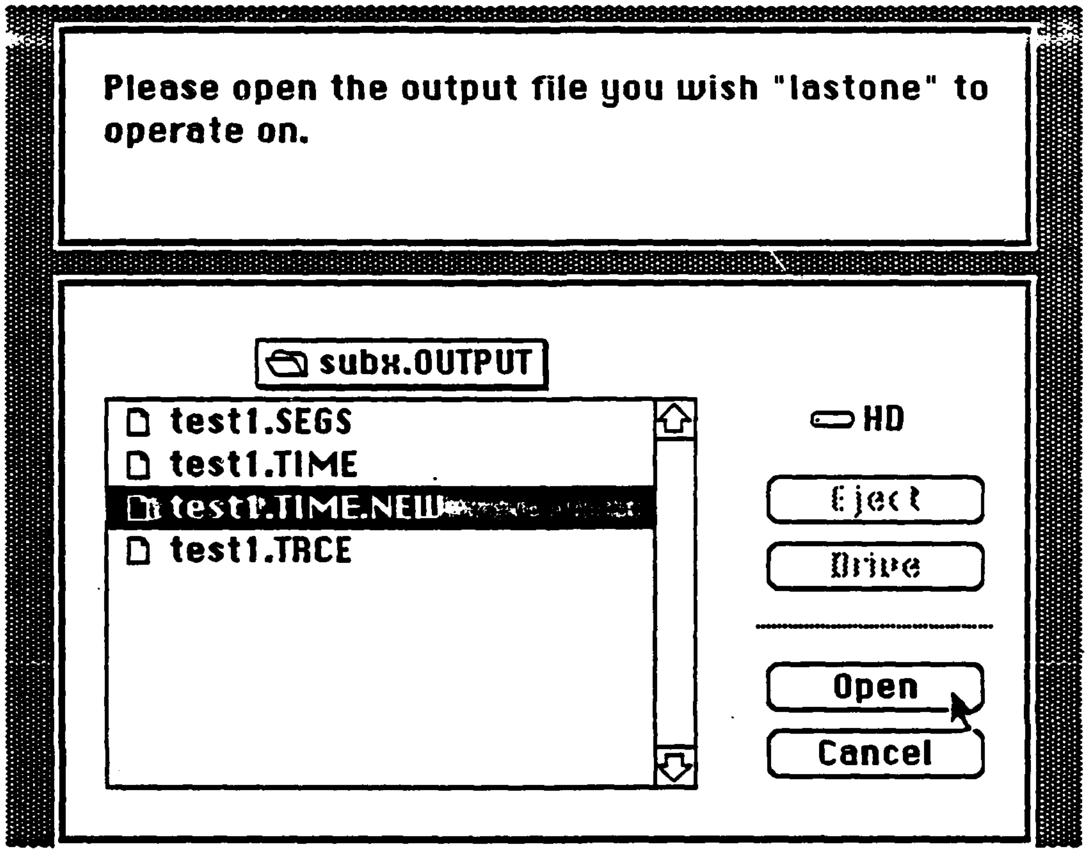
RUNNING LASTONE

Before running **Lastone**, you should have done the following: (1) Inserted the disk with the **Lastone** application on it; (2) Inserted the disk with the subject's output folder on it.

To start, double click on the **Lastone** icon. The first dialogue (Figure 21) is a two-part dialogue requiring a two-part action. The program needs to know which folder (and which file within it) you want to run the application on.

First, open the subject's output folder by clicking on, as before, "sub3.output".

Second, open the edited results of **Convert** by double clicking on the file with the "has-been-edited" tag to it; in our example, it was called "sub3.time.new-ed" (see **Macwrite** manual for more information).

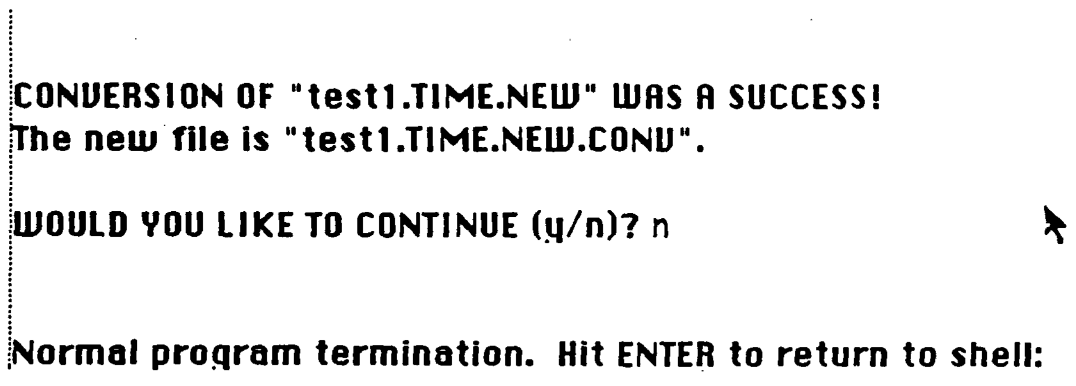


Please open the output file you wish "lastone" to operate on.

-- Figure 21 --

The program now begins its calculations and the screen goes blank - this is identical to what happens in **Convert.**

The final dialogue (Figure 22) informs you that **Lastone** was a success (if it wasn't, see below). Then it asks you if you would like to apply **Lastone** to another unique (edited) timefile. If so, type in "y" and hit return. If you are finished using **Lastone**, simply type "n" and hit return. The next line of the dialogue simply means, "hit return to get back to the desktop".



```
CONVERSION OF "test1.TIME.NEW" WAS A SUCCESS!
The new file is "test1.TIME.NEW.CONV".

WOULD YOU LIKE TO CONTINUE (y/n)? n


Normal program termination. Hit ENTER to return to shell:
```

-- Figure 2 2--

If for some reason you receive an error message or a bomb, return to the edited timefile. Did you save it as "text only"? Did you remember to give it a unique tag or name? And if so, did you select the correct file for use in **Lastone**? Is the disk bad? Did your data exceed one of the limits stated earlier? Was there enough space on the disk to save your file? Are you somehow using an old version of **Lastone** (dated before July, 1988)?

OUTPUT

The file resulting from running **Lastone** is saved into the output folder where the results of **Readit, Convert,** and **Macwrite** editing are saved. It will bear the name of the edited Macwrite file with an additional extension: ".conv". Thus, for example, if the file I submitted to **Lastone** were entitled "Sub3.time.new-ed", then the new file would be called "sub3.time.new-ed.conv". It should be obvious that the fewer keystrokes used, the better; with the constant addition of extensions, names can become unwieldy.

A real sample of the data created by **Lastone** and contained in the ".conv" file is shown in Table 3.

The top section of the table provides a brief explanation of the table headings (a further explanation follows Table 3).

The remainder of the printout contains actual data. For reasons of brevity, we show here the data for the first passage (named "13,2") only; in fact, data for all the passages read by the subject follows this initial information, in order read. In this example, the output folder and the ".conv" file were named after the subject number; therefore, as this is the data for subject 3, the output file is called "sub3.time.new-ed.conv."

SPECIAL NOTE: The output from **Lastone** corrects for the errors in the edited version of **Convert's** tWC and CUME columns because it counts the number of times a segment was read based on what's actually in the file. **Lastone** avoids the small editing errors in the cumulative columns by re-calculating the cumulative times based on the actual reading times in the tS columns of those segments in the edited file.

```
*********************************************************
THE FOLLOWING OUTPUT IS FOR SUBJECT "SUB3."
TIMES ARE IN VIEWING ORDER...
________________________________

#V.............Number of Segment Views.
#W.............Number of Words in Segment.
TWR............Total Words Read / Segment.
RT.............Total Reading Time / Segment.
RT/TWR.........Reading Time ÷ Total Words Read / Seg.
RT/#W..........Reading Time ÷ Number of Words in Seg.

*=============================================*

SEGMENT       #V   #W    TWR   RT      RT/TWR   RT/#W
13,2.SEG.1    1    20    20    10.033   0.502   0.502
13,2.SEG.2    1    17    17    37.650   2.215   2.215
13,2.SEG.3    2    17    34    35.950   1.057   2.115
13,2.SEG.4    3    10    30    24.166   0.806   2.417
13,2.SEG.5    2    15    30    25.450   0.848   1.697
13,2.SEG.6    3    19    57    21.316   0.374   1.122
13,2.SEG.7    3    22    66    79.650   1.207   3.620
13,2.SEG.8    3    21    63    38.883   0.617   1.852
13,2.SEG.9    3    19    57    37.917   0.665   1.996
13,2.SEG.10   3     7    21    13.217   0.629   1.888
13,2.SEG.11   3    15    45    79.233   1.761   5.282
13,2.SEG.12   2    14    28    16.950   0.605   1.211
13,2.SEG.13   2    12    24    14.466   0.603   1.206
13,2.SEG.14   3    19    57    29.900   0.525   1.574
13,2.SEG.15   2    16    32    22.217   0.694   1.389
13,2.SEG.16   2    11    22    13.467   0.612   1.224
13,2.SEG.17   2    26    52    25.717   0.495   0.989
13,2.SEG.18   2    20    40    83.384   2.085   4.169
13,2.SEG.19   1    13    13    10.850   0.835   0.835

Total Words In Passage =  313
Total Words Read =  708
Total Reading Time =  620.416
Total Reading Time ÷ Total Words Read = 0.876
Total Reading Time ÷ Total Words In Passage = 1.982
*********************************************************
```

-- Table 3 --

ABBREVIATIONS IN TABLE 3

#V - the total (cumulative) number of times a given segment was on the screen during the subject's session with a specific passage.
#W - the number of letters present in a given segment (taken from the data in the **Converted** timefile).
TWR - #V x #W, or the total (cumulative) number of words read for any given segment.
RT - the total (cumulative) amount of time spent viewing a given segment during the subject's session with a specific passage.
RT/TWR - the total reading time for a given segment divided by the total number of words read for that segment.
RT/#W - the total reading time for a given segment divided by the number of words in the segment.

DATA IN TABLE 3

If we now look at the data in table 3, we see the familiar columns with headers "SEG" plus the six terms listed above, for a total of 7 items. The data to which these headers apply covers information about segments and words within segments. The following provides a summary of the items and their interpretation.

Items 1-3:

These items label and provide descriptive data for each segment - taken from the **Converted** timefile. "Segs" labels the passage and segment number; "#V" tallies up the number of times the segment was viewed; and "#W" tells how many words are in the given segment.

Items 4-5:

These items provide data per segment, aggregated up to that point in the subject's reading process. "TWR" converts "#V" information into data about the total words read in a given segment; "RT" converts "#V" information into data about the total reading time spent on a given segment.

Item 6:

This item provides a "reading rate" measure, at the word level, for each segment. This measure can be used to determine whether and when subjects actually slow down or speed up their reading rate for the treatment segments or passages as against the untreated ones.

Item 7:

This item provides a "processing time" measure, at the word level, for each segment. This measure can be used to determine whether or not subjects actually spend more time processing the words in a particular treatment segment or passage as against the untreated ones.

SUMMARY INFORMATION FOLLOWING DATA IN TABLE 3

After the data for each passage comes a 5-line summary which provides nearly identical information to that explained above, only summarized at the PASSAGE level. The meaning of each measure is written out, and in any case mimics that above. It can be used to detect any differences between passages, or to get a baseline reading rate/processing time per word for a given set of subjects.

Taken together, the data and summary data in table3 provide rate, processing, and cumulative information at the word, segment, and passage level, and provide the type of data useful for further statistical inquiry.

It is strongly recommended that the results of **Lastone** be checked for validity as well as completeness. The completeness check can be done via **Viewdata** (see manual); you might as well print out the results since you will probably need hard copies to work from.

The first validity check should compare hand-done computations of the data to those produced by the program. Later validity checks would be required any time you create new passages, change the unit of measurement, or, naturally, if you modify any of the several applications which feed into **Lastone**.

VI. THE APPLICATION VIEWDATA

PURPOSE

Viewdata was created to allow quick and easy access to non-text documents otherwise tiresome to open through word-processing programs. The non-text documents may be viewed on the screen or printed.

CONSTRAINTS

(1) **Viewdata** can be used on text-only and regular text files. It cannot be used to look at program code or applications.

(2) **Viewdata** can only be used to look at and print files. It cannotbe used to EDIT any files.

RUNNING VIEWDATA

Before starting, you should have: (1) Inserted the disk on which the **Viewdata** application is located; (2) Inserted the disk on which the file to be viewed resides; (3) Have already done any editing or conversions of the file of interest.

To begin, double click on the **Viewdata** icon. Immediately a standard Macwrite-style dialogue window appears, offering you a choice of files which may be opened on all disks available. Simply open the file of interest. The screen will go blank, then the frame will appear, and shortly, the testfile contents will appear on the screen. You may either print the file or scroll through it.

To leave **Viewdata**, you must click "finished". If you want to look at a different file in **Viewdata**, then simply click on "new". The first dialogue window will reappear, this time showing the contents of the folder you were last using. To get out of the folder, simply click the "drive" button and you will return to the familiar choice menu.

OUTPUT

There is no file output resulting from the use of **Viewdata**. Rather, **Viewdata** can be used to check or to print out the data resulting from other applications.

Distribution List

Dr. Susan E. Chipman
Office of Naval Reserve
Cognitive Science Program
800 North Quincy Street
Arlington, VA 22217-5000

Defense Technical Information Center
Attention: T.C.
Cameron Station, Bldg. 5
Alexandria, VA 22314 (12 copies)

Library
Naval Training Systems Center
Orlando, FL 32813

Library, NPRDC
Code P201L
San Diego, CA 92152-6800

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Lynne Reder
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213