Error-information in tutorial documentation: Supporting users’ errors to facilitate initial skill learning

ARD W. LAZONDER† and HANS VAN DER MEIJ‡§
Department of Instructional Technology, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

(Received 22 November 1993 and accepted in revised form 13 January 1995)

Novice users make many errors when they first try to learn how to work with a computer program like a spreadsheet or wordprocessor. No matter how user-friendly the software or the training manual, errors can and will occur. The current view on errors is that they can be helpful or disruptive, depending on the extent to which they are controlled in the learning process. This study examines one of the ways in which such error control can be brought about, namely by investigating the design and role of error-information in a (tutorial) manual. The error-information was designed to support the detection, diagnosis and correction of errors of novice users, and it was based on a general model of error-handling. In an experiment a manual that contained ample error-information was compared to a manual in which there was hardly any error-information. The outcomes showed that the presence of the error-information in the manual helped subjects perform better during practice as well as after practice. Among others, these subjects completed training faster and showed superior corrective knowledge and skill after practice, in addition to having acquired the same level of constructive skill. The discussion addresses the compensating roles of support for error-handling on screen and on paper.

1. Introduction

Users’ first impressions of a computer program are vitally important for initial acceptance, for productivity and for continued use of that program. People persevere with a program for only a limited amount of time. If during this “honeymoon” period too many problems, mistakes and misunderstandings occur, the program is rejected and left to collect dust (Booth, 1991).

Software companies are, of course, working hard to prevent trouble and in case problems do arise, they try to help users deal with them. In doing so, they usually turn first to the program. That is, they rebuild certain aspects of the user-interface, and create or revise built-in support systems such as a help facility, and system messages. It is hoped that redesigning the software and the programmatic support for problems overcomes most of the users’ misunderstandings and errors. However, because the present technical know-how is not yet that advanced, this an optimism that is often not substantiated (Baber, 1991).

Even with relatively user-friendly programs such as word-processors considerably high failure rates have been found to occur. For example, Carroll and Carrithers

† Authors new address: E. J. Consultancy, P.O. Box 792, 7600 AT Almelo, The Netherlands.
‡ Author to whom reprint requests should be addressed.
§ Authors are in alphabetical order and no ranking is implied.
(1984) discovered that first-time users spend 25% of their time on dealing with errors. Other studies suggest that this percentage may even be as high as 50% (e.g. Card, Moran & Newell, 1983; Arnold & Roe, 1987; Graesser & Murray, 1990). Since many of the users' errors are not prevented by the program, nor covered by the programs' error support systems, the task of helping users in error-recovery still often comes down to the manual.

Unfortunately, most training manuals do not give users adequate support in dealing with errors. Manuals concentrate on teaching users what they can do with the software and how to do it (i.e. develop constructive knowledge and skills) without simultaneously training them how to undo the things that have gone wrong (i.e. develop corrective knowledge and skills). In addition, the literature on computer documentation gives only limited insights into the various ways in which error-handling can be supported. Little is known about how to present error-information in manuals. In this paper we therefore address this issue in depth, focusing on the design of tutorials for first-time users.

In designing paper support for error-handling at least two issues must be addressed. The first concerns the content of the error-information. What information do novice users need? The second issue concerns the presentation of the error-information. When and how should it be presented? Are there, for example, some rules of thumb that writers can apply to predict where problems are likely to arise and where help may be needed?

We deal with these two issues in the following sections. We investigate first how people generally deal with errors. This yields a description of the main error processing activities of users (i.e. detection, diagnosis and correction). We then go on to describe the principles for designing error-information in manuals, attending to content ("what") as well as presentation ("when" and "how"). Thereafter, an experiment is reported in which a manual with error-information designed according to the outlined principles is tested against a manual without such information.

2. A general model of error-handling

Several authors have suggested that adequate error-handling depends on what users do for detection, diagnosis and correction (Curry, 1981; Brown, 1983; Jelsma & Bijlstra, 1990; Reason, 1990; Wärn, 1991). Often, these processes are presented in a stage-like model, beginning with detection, continued with diagnosis and ending in correction, suggesting that error-recovery is a linear process. In reality users' error-handling may not always proceed in this fashion. But the model is a good starting point from a designers' perspective because it offers a solid basis for identifying the major processes in error-handling and hence provides valuable information on how to support these. In addition to describing the three main error-handling processes, we will briefly deal with the support programs (may) offer (i.e. error-messages, help system and system state information) in order to illustrate more clearly what paper support can or should do.

2.1. DETECTION

Users must first notice that they have made a mistake before they may initiate an attempt to correct it. Error detection is therefore conditional to any error-handling. According to Allwood (1984) detection may be triggered in two ways: internally or
externally. Internal triggering depends purely on reflections of the user leading him or her to the conclusion that something is amiss. In case of external triggering, some information on the screen or the manual onsets these reflections.

Internal triggering is said to occur when a user feels that he or she has done something wrong, but there is no visible cue to confirm this notion (yet). The user may, for example, feel insecure with the selected method, the command(s) or its execution. This state is typically signalled by wonderment questions such as “Did I do this right?” and “I wonder, have I completed all the necessary steps?”. In addition to raising doubts or uncertainty, the user must also come to the conclusion that an error has been made. He or she must classify an action as erroneous (and, hence, in need of repair).

External triggering is said to take place when a cue from the program or manual sets the user wondering about the presence of a mistake, or when such a cue directly tells users that a mistake has been made. External triggering can come about through three kinds of cues: program messages, program states and information from the manual. Here we will discuss only the programmatic support. A discussion of the paper support will be deferred until the next section.

Program messages usually alert users to both definite and possible mistakes. Some of the messages from programs clearly tell users that a mistake has been made. For example, messages like “Syntax error”, “Printer not selected”, “Wrong version number for dictionary xxx” signal the presence of a definite mistake. Other messages from the program are less direct, merely alerting users to a possible mistake. These messages allow for an interpretation that can go either way: the action may have been correct or incorrect. The message “Name already exists. Replace? No (Yes)” is a typical example, calling attention to an outcome that users may or may not want to achieve.

The alerting effects of program messages on users partly depend on their placement. Some programs present their messages in the middle of the screen, superimposing them on whatever screen is active at that moment. Such messages can be hardly ignored or not seen by the user. Other programs present their messages on a less prominent position (e.g. the bottom). For these programs there is a higher chance that users will not perceive them; these messages may be noted, or they may not be. The latter type of program messages displays may thus require additional cuing from the manual in order to be effective error-detection devices.

External triggering can also occur when something on the screen sets users wondering. For example a user of WordPerfect 5.1 who wants to put a word into italics and chooses the outline-option may note that the color code in which the word is presented on screen differs from that of words set in italics. This may prompt the user to choose the REVEAL CODES mode to examine this error. In general, it is probable that users of non-WYSIWYG programs profit more from information in the manual helping them detect incorrect program states that do users of WYSIWYG applications. This does not mean that the paper help is superfluous in WYSIWYG. Here too users may need to consult REVEAL CODES. For example, they may need to do so to check on the precise definition for a style. In addition, for novices some mistakes can be almost as elusive in WYSIWYG programs as in non-WYSIWYG programs (e.g. incorrect cursor movements).

The program's help system plays no role in error detection. Users deciding to consult help must already have come to the conclusion that something is amiss. At
best, program help can provide information that will secure users that an error has been made.

2.2. DIAGNOSIS

Diagnosis involves two activities: getting to know the exact nature of a mistake, and getting to know the action(s) that led to it. Occasionally, users can easily see or infer what is wrong and what caused the mistake. For example, after misspelling the name of a file and upon reception of the message "ERROR—FILE CHATPER1.TXT not found" the obvious conclusion is that the characters t and p have been reversed. In many cases, however, these diagnostic activities are very difficult for the novice user (see McKendree, 1990).

Fortunately, neither activity is conditional to error-handling. Users may start correcting a mistake immediately after its detection. For example, after a program message signals the presence of an error, some users will immediately select an undo option without pausing to reflect on the nature or cause of the error.

Knowing the exact nature of the mistake can be vitally important for the chances of success in correction, however. For this reason program messages often briefly indicate what is wrong: "bad name field", "disk error", "format file cannot be opened", "illegal variable name", "runtime error in line 327". Unfortunately, information about the cause(s) of errors is absent in many program messages. Thus giving users insufficient information for diagnosing mistakes. Especially for novice users this may obstruct error-recovery since these users often need more information to find out what is wrong and what may have caused it. As a result, users can fall back on only two external resources: the help system or the manual.

Unfortunately, good help systems for supporting diagnosis are rare. For example, in a study on the quality of help systems, Duffy, Mehlenbacher and Palmer (1992) found very low ratings for diagnostic support. On a scale ranging from no support to excellent support (0–100%) they found an average rating of 23% (for IBM) and 39% (for Macintosh) for the support help systems offered in "formulating the problem". Allwood and Kalén (1993) also attest to the poor quality of help systems as they report about a patient registration system in which no help was available on 93% of the occasions when users needed it.

In short, the burden of helping users diagnose mistakes often comes down on the manual. This may be more of an advantage than a disadvantage because navigation is easy, reading is comfortable, (re)reading is independent of the presence of an actual mistake, and more information can be presented on paper than on screen. Good diagnostic support in a manual should agree with what users see on the screen, it should point out the nature of the mistake and indicate possible causes (see Figure 1).

2.3. CORRECTION

Correction contains four kinds of user activities: (1) goal setting, (2) selection of the correction method, (3) planning the execution method, and (4) physical execution of the corrective actions.

The user begins with selecting a (repair) goal (Card et al., 1983; Allwood, 1984; Arnold & Roe, 1987). The top level goal is obvious: the gap between the present
and the desired outcome must be bridged. The user may break down this overall goal into a number of sub-goals (e.g. Frederiksen, 1984; Anderson, 1985; Rasmussen, 1986). For example, the goal “correct a typo” may be subdivided into the sub-goals “move the cursor”, “delete the incorrect text” and “type the correct text”.

Next, the user decides on following a corrective or a (re)constructive method. In a corrective approach users really make changes in the document that correct the mistake(s). For example, they may remove obstacles that are blocking a newly chosen option, or they may undo the error-state in order to get back to the situation right before the error. Users can also opt for a (re)constructive method, meaning that they simply try to perform the constructive actions again. By paying more attention, they hope to do it right this time.

Both approaches have certain advantages when compared to one another. Corrective methods are strong methods. They work for many types of error. These methods are, however, also difficult to learn. They are situation-specific and often require many different actions. (Re)constructive methods are easier to learn, but they are also weaker. They can remedy fewer error types than corrective methods can. In addition, they may leave behind some information (e.g. uncorrected hidden codes) that can affect later actions, or they may lead to a loss of information.

In many situations, however, both approaches are equally effective. For example, when a new line spacing code is placed before the old one, the user can correct the error by deleting the old code (i.e. a corrective method), or by moving the cursor to the right place and inserting the new code again (i.e. a (re)constructive method). When both methods work, method selection is handled by selection rules that allow the user to choose between them (see Card et al., 1983).

Third, the selected method is planned for execution. It is translated into a physical action-sequence. The user selects the commands that will be used and determines in which order they will be executed.

The last action in the model is the execution of the commands. This step plus the previous one may be executed simultaneously. That is, users may not plan the whole action sequence in advance but plan and execute step by step.

3. Principles for designing error-information in manuals

The general error-handling model describes a user’s actions in dealing with an error. To adequately support these actions, the training manual should incorporate error-information that parallels this recovery process to a certain degree. In the next section we will deal with the main principles for designing the content (“what”) and presentation (“when” and “how”) of error-information in manuals.

3.1. CONTENT

Error-information should specify explicitly when an error has occurred, and what must be done to get out of the error state (Lang, Lang & Auld, 1981; Allwood, 1986). Moreover, as novice users often find it difficult to know the exact nature and cause(s) of the error, and because users are at that time particularly receptive about explanations that can help advance their knowledge of the program (Van der Meij
& Carroll, 1995), it should also assist them in diagnosis. Good error-information should therefore contain: (1) a characterization of the system-state supporting the detection of the error, (2) conceptual information on the nature and likely cause(s) of the error, and (3) action statements for correcting the error (Lang et al., 1981; Mizokawa & Levin, 1988; Roush, 1992).

Among others, some fine-tuning between the error messages on the screen and the error-information in the manual may prompt variations. For example, when an error message displayed at the bottom of the screen informs users exactly about what went wrong and how the mistake can be corrected, the error-information in the manual needs merely to attend users to the screen (i.e. support detection). In these situations the user still gets complete information, but it is distributed over two different external sources. In other situations too there may not be a need to give information dealing with every activity in detection, diagnosis or correction. For example, for simple typing errors it may suffice to alert users to their presence and then give corrective information. In contrast, mistakes caused by a wrong analogy (e.g. typewriter analogy) may need more explanations about their nature. Not every error can or should thus be addressed in the same way. Uniformity in the "what" of a user's error-handling processes supported in the manual should not be the goal.

One of the most important issues in designing error-information that is adaptive to a user's actions concerns the treatment of different types of error. A common classification of errors is that into semantic errors, syntactic errors and slips (sometimes referred to as typing errors) (e.g. Douglas & Moran, 1983; Norman, 1983; Lewis & Norman, 1986; Reason, 1990).

Semantic errors are mistakes that arise out of the use of a method that is inappropriate in a particular situation. For this reason, semantic errors are intentional mistakes. The users' choice of method is incorrect in the given situation, it cannot achieve a given goal. For example, the user may select "Create Horizontal Line" to try to underline a word. In case of syntactic errors and slips, the user's intention is adequate, but the performance is deficient. When a correct command is carried out improperly, it is called a syntactic error. A typical example of a syntactic error in WordPerfect is ending the search mode by pressing the ENTER key instead of the F2 key. Slips are small errors at the keystroke level (e.g. mistyping the word in the search mode). These different error types require a different handling of the error-information.

Information in support of error-detection often must be more detailed for semantic errors than for syntactic errors or slips. Users often find it hard to detect semantic errors (Lewis & Norman, 1986; Rizzo, Bagnara & Visciola, 1987; Frese & Altmann, 1989). One of the reasons for this is that users believe they are following a correct method. Often their belief is contradicted only after they have completed all of their actions. External triggering should therefore not only clearly signal the error, it should also be very well-timed, making it unambiguously clear to users that they have made a mistake before they are tangled up so much in their error that it becomes too difficult to recover. In contrast, general descriptions of the error-state are often satisfactory to help users detect syntactic errors and slips. For these errors the discrepancy between an intended outcome and the actual outcome often can be observed immediately and easily by looking at the screen.

Information to support error-diagnosis also should vary for the different error
types. Diagnostic information about the nature of the mistake can be especially useful for semantic errors when it includes information about selection rules. That is, since the users chose a method that was inappropriate for the situation at hand, information that will inform them when the method applies can contribute to a better handling of it in future (and prevent the same mistake).

Diagnostic information concerning the cause of a mistake can be especially informative for syntactic errors and slips because the error-states resulting from these mistakes are often open to multiple interpretations. For these errors knowing what went wrong requires that users reflect upon their actions and consider these in relation to the outcome. For example, when a screen remains empty after an attempt to retrieve a document, the mistake may have been caused by a typo in the name of the document or by a syntactic error in one of the preceding menu choices. Novice users often need help here because they lack the thorough understanding of the software needed to make the correct inference.

Error-correction often does not relate to the type of error that is made. Slips can sometimes be as difficult to correct as semantic errors and recovering syntactic errors may also not be different in general. The content of the correction information is thus dictated by and large by what correction method applies, and, as discussed earlier, there is often a choice between a (re)constructive or a corrective approach. In general, however, it is not advisable to teach the use of a (re)constructive approach. Reconstructions can lead to sloppy work habits and may lead to hidden codes in a document that can frustrate work later on. In addition, they work only for a limited number of mistakes and the information about what to do is already presented in the manual anyway.

In presenting corrective information, the writer has a choice between the following approaches: teaching the use of a generic strategy or a specific one. Most programs nowadays enable the user to correct mistakes by following a generic recovery strategy or a specific one. For example, in most menu-driven software choosing the undo-option is a generic correction strategy that cancels the last action. In many MD-DOS programs, the ESC-key or F1-key serves this function. Specific correction strategies often depend on other actions than deselecting a particular choice. They often require several distinct actions. For example, correcting the mistake of activating a new document without closing another in WordPerfect 5.1 requires the following actions: "Press the F7 key and type an N twice. When the screen is empty, retrieve the new document." Generic methods have the advantage of being easy to learn and apply. They work for many errors and can be repeated over and over again. For this reason, they should be prominent in the manual and be treated early. When a generic method does not correct the error, specific correction information should be given.

3.2. PRESENTATION

In presenting the error-information to users the designer faces two important issues: when to give it and how to display it. The "when" issue has to do with the crucial factor of finding the right timing (or placement) for the information. The "how" concerns the signalling of the error-information.

Timing is crucial. One of the most important problems of novice users is that
unnoticed errors accumulate, getting them deeper and deeper into trouble. To prevent this, the right timing (or placement) of the error-information in the manual is critical. Error-information should be presented on the spot, directly after the actions it refers to (Carroll 1990; McKendree, 1990; Mory, 1992; Van der Meij & Carroll, 1995). Timely error-information allows for an early detection of errors which prevents errors from piling up, and, thereby, facilitates error diagnosis and recovery.

Some authors suggest there should be ample error-information in the manual (Carroll 1990; Thimbleby 1991; Lazonder, 1994a; Van der Meij & Carroll, 1995). In fact, our earlier research on this manner suggests that the inclusion of error-information at a rate of about once after every three actions may be just about the right frequency (Lazonder & Van der Meij, 1994). The frequent inclusion provides a safety net, supports more flexible and ambitious action-oriented learning, and at the same time can help deepen the users' understanding of how the program works.

The “1-on-3 actions” guideline is probably program specific. For other applications a different ratio may work better. Therefore, it is preferable to follow some more general heuristics. Our earlier studies suggest that error-information should be given when actions are error-prone and when mistakes are difficult to correct (Van der Meij, 1992; Van der Meij & Lazonder, 1993; Lazonder, 1994b; Van der Meij & Carroll, 1995).

Error-prone situations arise when automated behaviors do not apply (Booth, 1991), or when users are likely to call on a false analogy (Allwood & Eliasson, 1987). To a certain degree such situations can be foreseen by the writer. For example, in WordPerfect 5.1, an automaticity error was predicted and found when users wanted to search for a string of text. Instead of activating the search-command by pressing the F2 key, the users pressed the ENTER key, the default action to closing off nearly all menu-choices in the program. An instance of an analogy error occurred when a user substituted a “1” (the number one) with an “l” (the character l) in typing a filename. Although these characters are interchangeable on a typewriter, they have different meanings in word-processing.

Error-information should also be given when errors are difficult to correct (Van der Meij & Carroll, 1995). That is, when the error-state calls for a specific correction strategy involving many corrective steps.

Experienced writers can predict the “hot spots” in a manual, but they can do so only to a certain degree. Usability testing is always a vital strategy for finding out the right place (and frequency) of error-information in the manual, yielding indispensable information for the design of error-information (Carroll 1984; Schrifer 1989, 1992, 1993; Karat 1990; Mack, Lewis & Carroll, 1990; Thimbleby, 1991).

Of course, a clear textual and graphical presentation of error-information is indispensable as well. In keeping with the general error-handling model, a detection-diagnosis-correction format throughout the manual seems called for. Depending on the situation there may be a need for variations in content—such as for particular program messages—and style.

In addition, error-information has to be signalled to indicate its distinct nature and to facilitate recognition. For this reason, we propose to set the error-information in italics, not in the least place because italics tend to be read (a little bit) more slowly
1. If the text A/LETTER1.WP does not appear, you have forgotten to clear the screen. Press the F7 key and type an N twice to clear the screen as yet.

2. If the text Document to be retrieved: does not appear on the screen, you have selected the wrong command. Press the F1 key to rectify your choice.

3. If you have inserted the text at the wrong place, you have positioned the cursor wrongly before pressing the ENTER key. Remove the text again.

4. If you cannot entirely select the words "half a million dollars" you did not position the cursor at the beginning of this text before activating the block function. Press the F1 key to undo the block function.

5. If the text Drive not ready reading drive A appears, you have not inserted the diskette deep enough into the drive. Insert it again so that the button pops up. Then type a 1.

6. If the screen remains empty, you have probably made a typing error. Retype the name of the file and press the ENTER key.

Error type: semantic
Problem: detection and correction
Solution: the error-state is explicitly specified because the text of the file letter1.wp will appear on the screen anyway. Specific correction information is necessary: this is the only way to correct the error.

Error type: semantic
Problem: detection and correction
Solution: specific detection information to enable early detection. A generic correction method can be used to correct the error. Note that a specific diagnosis is fairly impossible here.

Error type: syntactic
Problem: diagnosis
Solution: detection and correction can easily be inferred by looking at the screen. Because the cut and paste function often is active in full-screen, diagnosis of the cause of the error is explicitly specified.

Error type: syntactic
Problem: diagnosis and correction
Solution: correctly positioning the cursor is frequently overlooked. Therefore, the cause of the error is explicitly included. Correction is a major problem as well. Unless the block function is switched off, the user cannot proceed. A generic correction method can be applied.

Error type: syntactic
Problem: detection, diagnosis, correction
Solution: detailed information for every error-recovery stage. A specific correction method is the only way out here. Also note that the full-stop at the end of the last sentence was omitted to prevent an accidental typing error.

Error type: slip
Problem: diagnosis and correction
Solution: specific information on diagnosis is included because it is not at all clear what caused the screen to remain empty. A specific correction method is the most efficient way to correct the error.

Figure 1. The left column shows six examples of error-information that are extracted from a manual for WordPerfect. The right column characterizes each example by specifying the type of error, the users' main problem(s) in recovering from that error and a rationale for the content of the error-information.

(Hartley, 1985), which may be helpful for having users execute the corrective actions accurately.

3.3. CONTENT AND PRESENTATION IN PRACTICE

Figure 1 presents a few examples of the error-information given in the minimal manual used in the experiment reported below. It shows that all error-information is presented within the same format: detection-diagnosis-correction. Different problems associated with each error (type) lead to variations in content and style.

To further explain how the principles regarding content and presentation of error-information were put into practice, two illustrative sections of this manual are shown in Appendix 1. These sections illustrate that the error-information has been made visibly distinct from the other information in the manual. Note also that the error-information is not piled up in one trouble-shooting section at the end of a method, but rather that it is given after particular steps of a method.
The manual included no error-information for task errors. Typical task errors for word-processors would be a badly structured text, grammatical and punctuation errors or font junk. During the usability test(s) we did find that users committed such task errors, but as these errors never disrupted task execution, they were not dealt with in the manual.

4. Investigating error-information

In a first study with a tutorial for novice users of a word-processing program (i.e. WordPerfect 5.1), we found no effects of error-information. Its presence in the manual did not improve the subjects’ performance (Lazonder & Van der Meij, 1994). Post-hoc analyses revealed some shortcomings that might account for this non-effect. For example, we found a ceiling effect on one of the corrective tests and we also discovered that the manual mainly supported syntactic errors and only a few of the semantic errors that users made. It was also in this study that we found out that the frequency of error-information was a bit low and that a fading technique (gradually decreasing the explicitness of the corrective actions) should be used sparingly. These detailed analyses of the users’ errors led to the more refined principles for the design of error-information described in this paper.

After making the necessary changes in the manual we conducted a pilot study (Lazonder, 1994b). The results of this study suggested that much more errors were now supported by the manual (approximately 40%). In addition, it was found that users frequently consulted the error-information and that it proved to be very helpful for error-recovery. The data from this study were used to improve the manual yet further.

This led to the present study in which we examine the effect of error-information on user behavior. In the experiment, half of the subjects were given a training manual with ample error-information (MM+) while the other half worked with a manual that contained hardly any error-information (MM−). The main hypotheses relate both to the learning activities and the learning outcomes. During practice, MM+ users are expected to commit fewer errors and to recover their errors faster than MM− subjects. Consequently, MM+ users are expected to require less time to complete practice. The inclusion of error-information should further be beneficial for the users’ scores on performance tests. More specifically, MM+ users should be more knowledgeable and skilled in detecting, diagnosing and correcting errors. In addition, we expect that they also develop better constructive skills.

5. Method

5.1. SUBJECTS

Fifty adult volunteers (10 men and 40 women) participated in the experiment. They were recruited by means of an advertisement in a local newspaper. The subjects’ mean age was 36 (S.D. = 10.0). Their educational background varied from secondary education to university. All subjects had less than 100 h of computer experience and no experience with the experimental software.

The subjects were randomly assigned to one of the two experimental conditions. There were 25 subjects in the MM+ group and 25 subjects in the MM− group. Checks on the random allocation to conditions showed the two experimental groups
to be essentially equivalent with regard to age, sex, educational level, intelligence, typing skill and prior experience with computers.

5.3. MATERIALS

5.2.1. Technical equipment
The experiment was performed on an Olivetti 286 personal computer with the menu-driven version of WordPerfect 5.1. WordPerfect’s help-function was disabled to restrain subjects from using this type of programmatic support. A registration program was installed on the computer. It generated a log file of all of the subjects’ actions. Whenever a key was pressed, time and keystroke were recorded.

The experimenter used the ERR-system (Error-Recovery Registration System) to log a subject’s actions in dealing with errors. This system is completely mouse-controlled and runs on an Apple Macintosh computer under HyperCard. By clicking icons, the experimenter can record when (a) an error is made, (b) an error is detected, and (c) the error-state is ended. When a given icon is clicked, additional information (e.g. type of error, quality of the solution) can be entered for that measurement. The clock time of both computers was synchronized to allow for cross-referencing between log files. Only measures with satisfactory inter-observer reliability scores (>0.70) were used in the experiment. Measures with scores below 0.70 were reassessed by checking the subjects’ log files.

5.2.2. Manuals
Subjects received a manual (MM+ or MM−) and a training diskette. Both manuals were so called minimal manuals, designed especially for the experiment and refined on the basis of several pilot tests (see Lazender, 1994b). The two manuals differed only with regard to the error-information. The MM− contained no error-information at all. In the MM+, error-information was presented frequently (i.e. 45 times). In all, over 35% of the MM+ consisted of error-information, designed according to the heuristics that were detailed above. The different types of error were equally addressed by the error-information. In the MM−, error-information was presented only once, namely in the first chapter where the usage of the “Undo” key F1 was explained. A more detailed description of the MM+ can be found in Carroll (1990), Lazender and Van der Meij (1993), Van der Meij and Lazender (1993) and Lazender (1994a).

5.2.3. Tests
Subjects’ intelligence was assessed by means of a standardized intelligence test (Raven, 1986). Three tests were used to assess learning outcomes.

A constructive skill test measured subjects’ constructive skill. It consisted of nine tasks. All tasks addressed elementary word processing tasks trained during practice (e.g. changing the line spacing, underlining words). The test measured performance success and time on task. It also gave some data on errors and error-handling. However, because the subjects’ own errors were uncontrollable by the experimenter and because all tasks were practiced, these data may give only a limited view on the error-handling capacities of the subjects.

A better impression of the subjects’ capacities in dealing with errors can be gained from the results on the two correction tests assessing respectively corrective
knowledge and skill. The corrective knowledge test was a paper and pencil test, consisting of nine items (five retention items, four transfer items). Each item presented a word processing task and a screendump, showing the result of the actions that were performed to accomplish that task. For each item, subjects had to mark all possible errors (i.e. detection). In case of an error, they also had to specify its most likely cause (i.e. diagnosis) and a way to correct it. The corrective skill test was performed on the computer. Subjects had to detect and correct eight errors (five semantic, three syntactic) in a task document (three retention items, five transfer items).

5.3. PROCEDURE

All experimental sessions took place in a quiet room. Each session lasted one day, with a maximum of 8 h. During the first half hour subjects completed the intelligence test (personal data such as age, sex and computer experience were collected by telephone). The remaining 7.5 h were spent on word processing. There were short breaks for coffee, lunch and tea.

The subjects were instructed to work through the manual in their own way and at their own pace. They were told that the experimenter would offer help only in case of a computer breakdown, or when the subject was stuck for more than 10 min. In addition, they were asked to think aloud during practice. Thinking aloud was rehearsed by tying a bowline knot according to a set of written directions. In performing this task, subjects were asked to (1) read aloud the action steps, (2) explain their goals and actions, and (3) evaluate their actions and comment on the results. Subjects were thus trained in explaining the key activities in word processing, namely planning, execution and evaluation of (a series of) actions.

Next, subjects were seated at a small desk with the computer and the printer in front of them. They were given their manual (MM+ or MM−) and a diskette, containing all documents to be used in practice. The manual managed the learning process by alternating short explanations with many practical exercises. During practice, the experimenter sat at a table nearby to record the subject’s corrective actions, using the ERR-system.

After practice, the subjects were given the constructive skill test and the corrective skill test. A counterbalanced administration was used to control for order effects. After these tests, the subjects completed the corrective knowledge test. During all tests, the subjects were not allowed to consult their manual or to ask for help of the experimenter.

5.4. CODING AND SCORING

During practice, the following measures were scored: time, number and type of errors, the number of detected and corrected errors. Practice time was the time to read the manual and complete the training exercises. With this measure, a distinction was made between time spent on constructive and corrective actions. Errors were scored as semantic, syntactic or slip. For each error, detection and correction were scored on a true–false scale.

Constructive skill was defined by test time, performance success and the number of errors. Test time was scored as the time subjects’ required to complete the
constructive skill test. Again, the distinction between time on constructive and corrective actions was made. Performance success was indicated by the number of successfully completed items on this test. For each subject, the number and type of errors in constructive performance was registered as well. All of these scores were assessed by examining the documents on diskette and the subject’s logfile.

Corrective skill was defined by three measures: detection, diagnosis and correction. Detection was scored as either right or wrong. Diagnosis was scored on the following 4-point ordinal scale: (1) both cause and effect are wrong, (2) wrong cause, right effect, (3) wrong effect, right cause, and (4) both cause and effect are right. For corrective knowledge, a similar scale was used. The correction method was scored as one that (1) obviously does not try to correct the error, (2) attempts to correct the error, but is semantically and syntactically incorrect or incomplete, (3) is semantically correct, but contains one or more syntactic errors, and (4) is both semantically and syntactically correct. The inter-rater reliability scores for all corrective measures was satisfactory (Cohen’s Kappa ≥ 0.80). Correction on the corrective skill test was defined as the number adequately corrected errors.

5.5. DATA ANALYSES

The majority of the data were analysed, first, by means of MANOVA’s using type of manual (MM+ or MM−) as the independent variable. Univariate analyses followed after a (marginally) significant multivariate effect was observed. Mann–Whitney U tests were applied to analyze the ordinal data. Where appropriate, an effect size (ES) was computed (in S.D.s) to establish the magnitude of statistically significant results.

Due to a computer breakdown, incomplete scores were registered for three subjects. In addition, one subject (MM−) did not get to the corrective knowledge test and one MM+ subject did not complete the corrective skill test. The data for these subjects were excluded on an analysis-by-analysis basis, causing variable group sizes.

6. Results

6.1. LEARNING ACTIVITIES

The mean time subjects needed during practice is presented in Table 1. Overall, the

<table>
<thead>
<tr>
<th>Table 1 Mean learning time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Constructive†</td>
</tr>
<tr>
<td>Corrective‡</td>
</tr>
</tbody>
</table>

Note. Time in min.
† Time spent on constructive actions.
‡ Time spent on corrective actions.
1 p < 0.05.
MM+ users were significantly faster than the MM− users \( (F(2,46) = 3.22, p < 0.05) \). This (8 min) difference can be ascribed to shorter correction time. MM+ subjects spent 38% less time on dealing with errors \( (F(1,47) = 4.35, p < 0.05, ES = 0.53) \). This time-gain of MM+ subjects could have been caused by any of the following factors: fewer errors, better or faster detection, and better or faster correction. Each of these factors is analysed below.

The data indicate that the MM+ manual helped reduce the number of errors during practice. Overall, MM+ users tended to make fewer errors than MM− users \( (F(3,46) = 2.66, p = 0.06) \), suggesting that the error-information helped prevent them from committing certain errors during practice. The data from Table 2 show that this is mainly a matter of making fewer semantic errors \( (F(1,48) = 6.14, p < 0.05, ES = 0.64) \).

The presence of error-information also affected error detection. MM+ subjects were significantly faster in detecting errors \( (F(1,48) = 4.51, p < 0.05, ES = 0.52) \). They tended to be better at detecting errors as well \( (F(1,48) = 2.35, p < 0.10) \).

Error-information also had an effect on error correction during practice. As can be seen from Table 2, MM+ subjects corrected more errors successfully \( (F(1,48) = 10.28, p < 0.01, ES = 0.80) \) and they also corrected errors faster than MM− subjects \( (F(1,48) = 4.90, p < 0.05, ES = 0.60) \). The error-information in the manual thus seems to have helped the MM+ subjects during training on all relevant factors for error-handling, helping them avoid errors, detect errors faster and correct these better and faster.

6.2. LEARNING OUTCOMES

6.2.1. Constructive skill

The subjects' "regular" performance on word processing tasks was assessed with the constructive skill test. This test measured success, time on task, number of errors and error-handling. No effects on time, or performance success were found (see Table 3). Also no effects on error detection or correction were present. MM+ did make 18% fewer errors, however. In particular, they were found to commit significantly fewer syntactic errors \( (F(1,46) = 4.99, p < 0.05, ES = 0.58) \).

6.2.2. Corrective knowledge

A marginally significant main effect was found for error detection \( (F(3,45) = 2.50, p < 0.10) \). Manual type affected the detection of semantic errors \( (F(1,47) = 4.00, p = 0.05, ES = 0.48) \) as well as syntactic errors \( (F(1,47) = 6.04, p < 0.05, ES = 0.70) \). As Table 4 shows, for both error types the MM+ subjects performed better than did the MM− subjects.

By and large the same results were found for error diagnosis, for which a significant overall effect was found \( (Z = 2.02, p < 0.05) \). MM+ subjects proved to be more capable of diagnosing syntactic errors \( (Z = 2.75, p < 0.01) \). For semantic errors the quality of their diagnoses was only marginally better than that of the MM− subjects, however \( (Z = 1.37, p < 0.10) \).

The results on error correction again showed a significant overall effect supporting the MM+ subjects \( (Z = 1.70, p < 0.05) \). But only their performance on syntactic errors was significantly higher than that of MM− subjects \( (Z = 2.48, p < 0.01) \).
TABLE 2
Mean error-handling scores during practice

<table>
<thead>
<tr>
<th></th>
<th>MM+ M</th>
<th>S.D.</th>
<th>MM- M</th>
<th>S.D.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committed errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic‡</td>
<td>8.4²</td>
<td>5.5</td>
<td>12.7</td>
<td>6.7</td>
</tr>
<tr>
<td>Syntactic</td>
<td>8.8¹</td>
<td>5.0</td>
<td>11.6</td>
<td>6.1</td>
</tr>
<tr>
<td>Slip</td>
<td>2.8</td>
<td>1.8</td>
<td>2.8</td>
<td>1.9</td>
</tr>
<tr>
<td>Error detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.4²</td>
<td>0.3</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>Success‡</td>
<td>90.1¹</td>
<td>9.0</td>
<td>85.9</td>
<td>8.5</td>
</tr>
<tr>
<td>Error correction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>0.9²</td>
<td>0.5</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Success§</td>
<td>86.1³</td>
<td>10.7</td>
<td>74.7</td>
<td>14.2</td>
</tr>
</tbody>
</table>

Note. Time in min.
† Mean number of errors.
‡ % of detected errors.
§ % of successfully corrected errors.
¹,p < 0.10 ²,p < 0.05 ³,p < 0.01.

In short, the data for corrective knowledge give a fairly consistent pattern of superiority of the MM+ subjects. Advantages were found for detection, diagnosis as well as correction. The subjects proved to be especially knowledgeable about handling syntactic errors. For semantic errors the results were somewhat more varied. For slips no effects whatsoever were found.

TABLE 3
Mean scores on the constructive skill test

<table>
<thead>
<tr>
<th></th>
<th>MM+ M</th>
<th>S.D.</th>
<th>MM- M</th>
<th>S.D.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constructive‡</td>
<td>25.1</td>
<td>11.8</td>
<td>28.3</td>
<td>12.5</td>
</tr>
<tr>
<td>Corrective‡</td>
<td>8.1</td>
<td>9.2</td>
<td>8.1</td>
<td>7.7</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success§</td>
<td>6.2</td>
<td>1.5</td>
<td>5.4</td>
<td>1.8</td>
</tr>
<tr>
<td>Committed errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>5.2</td>
<td>4.6</td>
<td>5.8</td>
<td>4.3</td>
</tr>
<tr>
<td>Syntactic</td>
<td>2.5¹</td>
<td>1.7</td>
<td>3.8</td>
<td>2.2</td>
</tr>
<tr>
<td>Slip</td>
<td>1.0</td>
<td>1.1</td>
<td>1.0</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Note. The constructive skill test consisted of 10 items. Time in min.
† Time spent on constructive actions.
‡ Time spent on corrective actions.
§ Number of items successfully completed.
¹,p < 0.05.
Table 4

Mean scores on the corrective knowledge test

<table>
<thead>
<tr>
<th></th>
<th>MM+</th>
<th></th>
<th>MM+</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>S.D.</td>
<td>M</td>
<td>S.D.</td>
</tr>
<tr>
<td>Detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>3.91</td>
<td>0.9</td>
<td>3.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Syntactic</td>
<td>2.12</td>
<td>0.9</td>
<td>1.5</td>
<td>0.9</td>
</tr>
<tr>
<td>Slip</td>
<td>0.4</td>
<td>0.5</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>Diagnosis†</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>27.71</td>
<td>22.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic</td>
<td>30.33</td>
<td>19.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slip</td>
<td>25.8</td>
<td>24.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correction†</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>27.0</td>
<td>22.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic</td>
<td>29.93</td>
<td>19.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slip</td>
<td>26.0</td>
<td>24.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note. The corrective knowledge test consisted of 9 items.
† Mean rank scores, higher ranks indicate higher quality.
1 p < 0.10  2 p < 0.05  3 p < 0.01.

6.2.3. Corrective skill

There were no effects of manual type on detection. MM+ subjects did only slightly better than the MM− subjects (see Table 5). It is recalled, however, that during the skill test users can exploit the program’s warnings and error messages, as well as study the hidden codes in a document. Especially for detection this would seem to be an important factor. We will return to this issue in the discussion.

A significant multivariate main effect was found for correction ($F(2,46) = 4.35$, $p < 0.05$, $ES = 0.71$). MM+ subjects were more skillful in handling both semantic errors ($F(1,47) = 8.41$, $p < 0.01$, $ES = 0.79$) and syntactic errors ($F(1,47) = 5.23$, $p < 0.05$, $ES = 0.65$) than MM− subjects.

Table 5

Mean scores on the corrective skill test

<table>
<thead>
<tr>
<th></th>
<th>MM+</th>
<th></th>
<th>MM−</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>S.D.</td>
<td>M</td>
<td>S.D.</td>
</tr>
<tr>
<td>Detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>3.6</td>
<td>1.3</td>
<td>3.5</td>
<td>1.4</td>
</tr>
<tr>
<td>Syntactic</td>
<td>2.6</td>
<td>0.7</td>
<td>2.3</td>
<td>1.1</td>
</tr>
<tr>
<td>Correction†</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>2.21</td>
<td>1.1</td>
<td>1.3</td>
<td>1.2</td>
</tr>
<tr>
<td>Syntactic</td>
<td>1.82</td>
<td>1.1</td>
<td>1.0</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Note. The corrective skill test consisted of 8 items.
† Mean rank scores, higher rank indicates better error correction.
1 p < 0.05  2 p < 0.01.
The outcomes on the corrective skills test thus speak favorably for the presence of error-information as well, but only with regard to error correction. On detection no effects were observed.

7. Discussion

This study examined the effect of supporting users' error-handling in learning to use software. This support came in the form of error-information that was included in the training manual to assist users in their error-handling processes. Error-information was expected to have a positive effect on users' constructive and corrective knowledge skills during as well as after practice.

The outcomes of this study clearly support the conclusion that presenting error-information in a manual improves performance during practice. The data also showed significant gains for error-handling knowledge and skills on the corrective tests. In contrast, the outcomes on the constructive skill test were, except for error frequency, unaffected by this presence. In discussing these results, the effects of error-information on users' performance during practice will be addressed first. Then we will deal with the outcomes on the constructive skills test and consider the results for error-recovery on the corrective knowledge and skill tests. Finally, the advantages of paper and on-line support for error-handling are discussed.

MM+ subjects completed practice faster, a result that was ascribed to fewer errors and better error-handling (i.e. faster detection and better and faster correction). The fact that MM+ users were not faster at performing constructive actions may well be explained by the presence of error-information. Post-hoc analyses showed a positive correlation between consulting error-information when no error was made and constructive time \((r = 0.41, p < 0.05)\). In other words, subjects repeatedly used the error-information for exploratory purposes, thereby affecting constructive training time negatively.

The presence of error-information did not affect the subjects' constructive performance on the constructive skill test. This may be due to the fact that the two manuals presented the same constructive content. However, on error-handling only one of the anticipated effects was found: MM+ subjects committed fewer syntactic errors. The absence of any other effects on error-handling may well be explained by the fact that all of the items of the constructive skill test had been practiced (no transfer items were included in this test). This may have made most of the subjects' errors trivial and easy to correct. A more intriguing explanation, however, is that the availability of error-handling information on screen has affected these outcomes.

There is, first, the presence of WordPerfect's monitoring options that give users a print preview or enable them to analyse the hidden codes within a document. Post-hoc analyses showed that users consulted these options frequently (i.e. 82% of all users consulted these at least once) and that these consults correlated positively to the detection of syntactic errors \((r = 0.31, p < 0.05)\). This, in turn, also had a positive effect on the number of correctly solved items \((r = 0.31, p < 0.05)\). In short, by consulting the program's monitoring options, users were supported in detecting and correcting errors.

Yet another factor that may have affected the subjects' performance on the constructive skill test are the program's system cues. System cues provide users with
valuable information on the occurrence of an error. Cues like "ERROR--FILE CHATPER1.TXT NOT FOUND", "Exit WordPerfect? No (Yes)", or "Block on" may prompt users to review their goal, their solution method, or the execution of some action step(s). Some system cues may even block task progression. The present study was not designed to examine this effect, but it is of interest to address this issue in future studies to assess the effect of this on-line error-information.

The presence of the error-information had a considerable impact on the subjects' error-handling capacities after practice, as shown on the two corrective tests. On the corrective knowledge test significant gains were found for detection, diagnosis and correction. The corrective skill test likewise revealed several positive effects on correction. The inclusion of error-information in the manual thus led to important improvements in the subjects' corrective knowledge and skill.

On the corrective skill test no difference in error-detection was found. We believe that this was due to the subjects' use of WordPerfect's monitoring options. Post-hoc analyses supported this stance, showing (again) a positive correlation between consulting these options and error detection ($r = 0.28, p < 0.05$). In contrast, on the corrective knowledge test the presence of error-information did affect error-detection. This test is critical in that it is the only one in which users cannot profit from cues that appear on the screen. It assesses unprompted recall; users can rely only on what knowledge they have developed of the program in dealing with a (possible) error. This test thus seems to indicate that the error-information also helps users to develop a better understanding of the program.

The present study suggests that creating good support for error-handling by users is not simply a matter of improving the program so that there is no longer any need for including error-information in the manual. Instead the two can, and should, complement one another, thereby offering (novice) users the best of both worlds.

The interactive promises of programs offer unique possibilities for negotiating meaning with the user and for giving feedback on users' actions—a support paper manuals can never offer as effectively. Error-information can also be made less salient on screen than on paper. That is, it can be presented upon the user's request only. In contrast, error-information in the manual can at best be made as unobtrusive as possible, facilitating skipping when not necessary. Moreover, on screen error-information can be "build-up". It can be designed so that it provides only a minimum amount of support in the beginning, which can then be elaborated upon after a user request for more information. The general model of error-handling presented in this paper suggests what content elements one could consider in this respect. For example, the presentation of diagnostic information may well be deferred until later. Obviously, such a "build-up" of information is not possible in a paper manual.

There are, however, several advantages to presenting error-information on paper as well. The most important one is that error-information presented on paper taxes users less than when this information must be accessed on screen. Activating on screen error-information requires (some) computer skills. Among others, users need to be able to activate the help option and navigate through its index or table of contents. On paper this is easy, especially when the manual is well-organized and thin. Consequently, an on paper support for error-handling seems best suited to
novice users, an audience that often does not possess the necessary computer skills. And even when they do possess these, the required actions are likely to overload their processing capacities which may cause them to commit additional errors.

A similar advantage of paper to on screen information is found when error-information is imposed on the user. That is, when error-messages supporting detection, diagnosis and correction of a possible error are generated by the software. Again, basic computer skills are required to benefit from this kind of support (e.g. to close the window in which the message is presented). First-time users often do not possess these skills yet. Some software-driven error support might thus be less helpful than when such support is provided on paper.

Another advantage of paper support for error-handling concerns its unique capacity for dealing with semantic errors. Just the fact that such errors lie at the user's level of the intention make them unfit for software-driven detection. Only paper support with its expressed linkage between goals and actions (especially strong in tutorials) can support the early detection of these errors and thereby increase the chances for recovery.

Yet another reason why error-information in the manual is useful relates to its role as a feedback mechanism and as a stimulus to monitoring performance. The error-information in the manual always directs the users' attention to the screen to check whether an error has occurred (hence the "if...then" construction). Observational data from the ERR-system revealed that users frequently consulted the error-information for this purpose. In fact, over 70% of the error-information in the MM+ was used for monitoring. Subjects used it to check the system state with the error-state in the error-information. The high consultancy rate suggests, first, that the information served as feedback and, second, that it prompted frequent monitoring.

In conclusion, there are important advantages to be gained from supporting user errors both on paper and on screen. This article has outlined some of the ways in which this can be achieved and how the two might "join forces" in assisting the (novice) users.

References


Paper accepted for publication by Associate Editor, Dr S. Treu.
Appendix 1

Typical sections from the training manual (MM+)

**Ending WordPerfect**

You should not turn the power off without ending WordPerfect first.

1. Press the ALT key
2. Press the ↓ key

The menu under the word File appears.

3. Put the brightened bar on the command EXIT by pressing the ↓ key
4. Press the ENTER key

*If the question Save document? Yes (No) does not appear, you have selected the wrong command. Press the F1 key to rectify your choice.*

5. Answer the question by typing an N

*If the text Document to be saved: appears, you have pressed the wrong key. Press the F1 key and end WordPerfect again.*

WordPerfect asks you a second question.

6. Answer this question by typing a Y

*If WordPerfect remains on the screen, you have pressed the wrong key. End WordPerfect again.*

You are back to where you started. The text C\> appears on the screen. You have ended WordPerfect.

---

**Retrieving a document**

Before you can retrieve a document from disk, you must always clear the screen first.

1. Go the the menubar and choose the command EXIT
2. Press the ENTER key
3. Answer both questions by typing an N

You have cleared the screen.

*If there is still text on the screen, you may have pressed the wrong key. Press the F7 key and type an N twice to clear the screen as yet.*

4. Go the the menubar and choose the command RETRIEVE
5. Press the ENTER key

*If the text Document to be retrieved: does not appear, you have selected the wrong command. Press the F1 key to rectify your choice.*

6. Type MANUAL.TXT
7. Press the ENTER key

The document MANUAL.TXT appears on the screen.

*If the screen remains empty, you have probably made a typing mistake. Retype the name of the document and press the ENTER key.*