2004

A menu driven, user friendly interface to UNIX

Mary Hayward

A MENU DRIVEN, USER FRIENDLY INTERFACE TO UNIX*

Mary Hayward

a thesis submitted to the faculty
of the School of Computer Science and Technology,
Rochester Institute of Technology
in partial fulfillment of the
requirements of the degree of
Master of Science

Approvals:

Peter H. Lutz (chairman)

Henry A. Etlinger

Warren R. Carithers

* UNIX is a Trademark of Bell Telephone Laboratories, Inc.

Title of Thesis: A MENU DRIVEN, USER FRIENDLY INTERFACE TO UNIX

I _____ hereby grant permission to the Wallace Memorial Library, of R.I.T., to reproduce my thesis in whole or in part. Any reproduction will not be for commericial use or profit.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dr. Peter E. Lutz for his continued interest and assistance in this thesis project, and for his guidance throughout the entire master's degree program.

Appreciation is also extended to Mr. Henry A. Etlinger and Mr. Warren R. Carithers for their assistance in the completion of this project.

I would like to thank by husband, Craig and my children who have supported and encouraged me in my educational endeavors.

ABSTRACT

A MENU DRIVEN, USER FRIENDLY INTERFACE TO UNIX

An interface was developed to provide an environment where an inexperienced UNIX user can work effectively within the UNIX operating system. A by-product of the interface results in the user learning a subset of commonly used UNIX commands. The interface incorporates a menu structure which presents the available UNIX functions to the user. When the user selects a function, the interface prompts him for the required arguments, echos the corresponding UNIX command and executes the command.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. Introduction

1.1. Demand for user friendly interfaces

> Man must become the prime focus of system design. The computer is there to serve him, to obtain information for him, and to help him do his job. The ease with which he communicates with it will determine the extent to which he uses it. ... To be effective, systems will have to be designed from the outside in [MART73].

Computer usage has increased dramatically over the past few years. We can expect an even greater integration of the computer into man's life, both in his work and home activities. As the number of computer users increases the characteristics of the user will diversify. Computer usage is no longer solely associated with programmers and technicians. Instead, individuals from all walks of life are actively using computers. It is no longer reasonable to expect a user to have computer training to become an effective computer user.

The man-machine interface is becoming a critical area in computer systems. The interaction between man and the computer gains significance not only in applications software but also in operating system design, programming

languages and documentation. 'In today's complex world, man and machine work interactively. The 'system' is a combination of both' [SIMP82]. An interactive computer system must be designed to meet the functional requirements of the user and yet remain efficient and easy to learn. A powerful functional capability has little value if it is so difficult that the user chooses to disregard it [MORA68].

The software a user interacts with can be seen as having two orientations: system-oriented software and user-oriented software. System-oriented software includes commands which perform file maintenance, commands for accessing utilities, and commands for use of assemblers, compilers and editors. It hides the many interface details from the user in order to make the system easier to access and use. The user must initiate the commands needed and remains responsible for the results he initiates. In contrast, user-oriented software is concerned with the user and his needs rather than the system and its needs.

The operating system is the interface between the hardware and the user. An individual who solely runs applications software such as payroll packages or computer aided instruction, will have minimal interaction with the operating system. His knowledge may be limited to the

commands for logging into his system and running his programs. For other users the interaction with the operating system will be of greater magnitude. For these users the dialogue between the user and the operating system is a significant factor in their productivity and enjoyment of the computer system.

1.2. Description of the UNIX operating system

The UNIX operating system was developed at Bell Laboratories by Ken Thompson and Dennis Ritchie. Thompson and Ritchie originally designed UNIX for their own use as an aid in software research. Word spread quickly regarding the power and flexibility of the UNIX operating system. 'The general sense of the word power is usually something like 'being able to do more with less effort.' Applied to software such as an operating system it translates into convenience of use" [SWAI82]. Research and academic institutions were interested in implementing UNIX. In 1973 Western Electric made UNIX available to universities and other nonprofit organizations. UNIX has since been updated by Bell Laboratories and other organizations [GRFE81]. UNIX growth has been rapid; according to some, it may even become the standard operating system for mini and micro computer systems [SWAI82].

UNIX is powerful and flexible without resorting to a complex system. An experienced programmer can quickly understand the entire operating system. The following UNIX design goals described by Greenburg provide a basis for a discussion of the strengths and weaknesses of UNIX:

> 1. Support a basic functionality within the operating system itself, relying on the normal user programs to provide sophistication. Features such as login/ logout and line printer queuing are implemented as normal user programs instead of operating system functions. This approach reduces the complexity of the system.
>
> 2. Generality - having a single method serve a variety of purposes. This generality enables one to program easily along with providing flexibility and extensibility.
>
> 3. Accomplish large tasks by combining several smaller tasks whenever possible. Again this design goal encourages a modular approach to complex programs and keeps the underlying operating system simpler [GREE81].

It is also beneficial to note several of the features of UNIX as described by one of its designers Ken Ritchie.

1. File structure

> The UNIX operating system is not tied down to the notion of a record. Each file is regarded as a featureless, randomly accessible sequence of bytes. The system conceals physical properties of the device on which the file is stored. File space is not preallocated and system calls to read and write have only one form.

2. Structure of the file system

"The UNIX file system is a recursive structure operating from a root directory. The root directory contains the names of files and subdirectories; the subdirectories contain names of files and other subdirectories, etc." [GREE81]. The structure has the form of a tree. It is inexpensive to implement and efficient in search time.

3. Input/output devices

UNIX has tried to remove differences between disk files and I/O devices including terminals. The user's terminal is important in inter-user communication and redirection of input and output to other files.

4. User interface

UNIX uses a 'shell' as a means of communication between the user and the operating system. The shell interprets user commands. The shell is not part of the operating system. It can easily be replaced with another shell program thus providing a new interface to the user. Individual command structure is simple and regular. Command arguments are separated by white spaces and symbols such as '<', '>', or '>>'. The shell parses the commands and passes them on as separate strings, thus the user is not required to learn a complicated syntax involving commas, semicolons and parenthesis. One of the most important aspects of UNIX programming is the concept of pipes. A pipe is an open file connecting two processes. The shell provides a notation for pipes that facilitates their use and enables interconnection of programs to be a commonplace activity.

5. Environment of a process

"New processes are created by the fork operation, which creates a child whose code and data are copied from the parent. The child inherits the

open files of the parent and executes asynchronously with it unless the parent explicitly waits for termination of the child." When the shell executes a command, each command becomes a separate process. Communication between processes is handled by sending "signals".

6. Reliability

UNIX reliability has generally been good, however, running out of swap space or an irrecoverable I/O error during swapping will cause the system to crash. A reliable system should not lose or corrupt files if the system crashes. UNIX does not take any special precautions to keep user files intact. UNIX is not very tolerant to hardware inconsistencies and problems.

7. Security

UNIX was designed in a open environment where security was not an issue. Consequently security problems do exist within UNIX. Security shells have been developed for applications that require greater security [SWAI82].

8. Use of a high-level language

UNIX is written primarily in the C language. Because the operating system is written in a high level language it is easy to modify it to meet an installation's unique needs. Of even greater significance is the portability that a high level language offers. Users are given the option of an operating system independent of the hardware on which it runs. "As the price of software rapidly out paces that of computers, the need to increase software productivity and reduce duplication of effort has become paramount" [GREE81]. Application programs can be written to run on diverse hardware configurations [RITC78].

1.3. Recluse nature of UNIX

Most UNIX users are enthusiastic and pleased with the positive features UNIX includes. However, there exists a common area of concern regarding the user interface and the ease of use. Ritchie has said:

> Both input and output of UNIX programs tend to be terse. This can be disconcerting especially to the beginner. The command interpreter does not remark loudly each time a program finishes normally, or announce how much time and space it took. Likewise, commands seldom prompt for missing arguments; instead, if the argument is not optional, they give at most a one line summary of their usage and terminate [RITC78].

The following examples illustrate the terse and often confusing interface of UNIX:

1. Often command names do not reflect the task to be accomplished; e.g., "cat" does not suggest by its name that it can be used to list contents of a file.

2. There are inconsistencies in abbreviations for commands; e.g., the four letter command "date" is not abbreviated but the four letter word "copy" is abbreviated to "cp".

3. Commands exist that produce output without labels or column headings; e.g., "who" returns a one line description of users currently logged into the system but uses no column headings.

4. UNIX tends to be silent; a user does not receive recognition from the system as to his ex-

istence unless he makes an error.

The terseness of UNIX may be frustrating to the new or casual user; however, the experienced user finds it desirable. Once an individual gains experience with UNIX he appreciates the user interface. He is able to communicate quickly and efficiently and is not bothered with excess noise sent to him by the operating system.

1.4. Statement of thesis project

The above discussions of the need for user friendly interfaces, the features of UNIX and the recluse character of UNIX are brought together in this thesis project. The purpose of the thesis project was to design and implement a user friendly interface to UNIX. The interface is a program, or "shell" that enables inexperienced UNIX users to function comfortably within the UNIX environment. It also teaches the user appropriate UNIX commands to complete his desired tasks.

CHAPTER 2

REVIEW OF USER FRIENDLINESS

2. Review of user friendliness

2.1. Importance of user friendly software

Advances in hardware development have been rapid. The computer of today is a vastly different machine from the early computers. The new technology becomes an integral part of daily activities. Software too has advanced over the years; however, there remains a void in guidelines for optimal man-machine interface. "We have been so bound by technology (removing obvious annoyances of existing interfaces and getting the bandwidth up, all within the bounds of economy), that we have not addressed substantive issues of man-computer communication" [ROBE81].

> Despite the current interest in user-computer interfaces the design of a good interface remains to a great extent an art, with much argument over guidelines and principles for interface design. Pertinent information, scattered throughout the literature of psychology, graphic design, linguistics, hardware design and under the general umbrella of computer science is only gradually being gathered together into survey publications for applications in computer science [BLFS81].

A common thread throughout the literature expresses a concern and realization that the man-computer interface is a fertile area for further research [BLAC81]. There is a desire to see interactive dialogue move from an art to systematic dialogue engineering [GAIN78]. Interactive systems need guidelines for dialogue enabling the system designers to avoid software that operates effectively for the designer but leaves the end user in a confused and frustrated state. The following discussion will look into the literature with the goal of better understanding man-machine interfaces. An emphasis will be placed on the principles directly related to the thesis project of the user friendly interface to UNIX.

2.2. Principles of human communication applied to man-machine dialogue

An interactive man-computer system involves a dialogue or communication between man and the computer system. "To be truly usable, a system must be compatible not only with the characteristics of human perception and action but more critically also with users' cognitive skills in communication, memory and problem solving" [CLAR81]. When human dialogue is examined, strategies and procedures are revealed. Often man-computer dialogue is designed incorporating a near-English vocabulary and syntax hoping the result is a user friendly interface. A near-English

vocabulary is not enough as it does not include the underlying strategies of human communication. Four underlying principles of human communication and how they are vital to man-machine dialogue will be examined [JONE78].

2.2.1. Expectation, prediction, and familiarity

When two people meet for the first time they incorporate past experience. Person A does not expect person B to bite him. Person C does not look at person D and then hide in a corner. Instead A, B, C, and D expect a normal introduction and greeting. It should be noted that a conversation will take place without consulting a manual describing the people involved. When we look at computer dialogue the opposite is true. Often the first thing a user encounters is a formidable manual which proceeds to show him how new and how different the computer system is; perhaps training courses covering use of the system will even be required. Human beings enter any experience with expectations which should be accounted for in system design.

Experimental research has affirmed the need for using prior real world knowledge during communication [BLAC81]. Man-computer dialogue should allow for these expectations and use them in a positive manner. Once an individual has used a particular dialogue technique, he will be able to

predict how a similar model will work. 'A good standard dialogue technology has the same impact as a good standard programming language - users can transfer experience from one situation to another" [GAIN78]. When a given technique or model for dialogue is adopted, the dialogue should consistently follow the model to take advantage of the user's predictability. If a user is accustomed to seeing error messages at the bottom of the screen he will expect feedback in that location. System designers can anticipate the users response and program accordingly.

A designer must carefully consider the naive user's past knowledge, determine if it is congruent with an experienced user's model, and use caution when the two models do not coincide. In 1979, Bott conducted an experiment which demonstrated a situation where a user's prior knowledge was detrimental to his understanding in a man-computer interaction. Bott presented naive computer users with instructions on how to use the print command in the text editor of the UNIX operating system. Each subject responded with his interpretation of each instruction after it was presented. When the subjects' were given the simple introductory line of 'How to print text', the responses were far from the experienced user's interpretation (displaying some written material on an output device such as a CRT screen). The subjects prior knowledge of

"print" referred to printing presses, "text" referred to a book and 'how' suggested a set of directions for doing a task. The subjects interpreted the line as procedures for printing a book on a printing press. In this instance the subjects' real world knowledge resulted in an incorrect interpretation [BOTT79]. Perhaps if real world knowledge had been considered when designing UNIX this type of misunderstanding could have been avoided. It must be recognized; however, that UNIX was not designed with the naive user in mind, but rather for Thompson's and Ritchie's personal convenience.

2.2.2. Implication

The set of interactions prior to a given statement make up the context of that statement. Human beings imply much from the context of a statement. The use of implication is seen in today's computer systems. The user is not surprised to encounter a different set of commands as he moves from the editor to the operating system. He realizes that he is in a different context or mode. It is reasonable to expect the user to make implications in a man-computer dialogue on the basis of context. However, a designer must be cautious when relying on the user's implications. "Consistency and compatibility are generally acknowledged to be basic ergonomic principles. A system which provides the user with a consistent

representation within a particular type of knowledge is likely tc be easier to learn and less prone to error than an inconsistent one' [BARN81]. Consistency in not always possible and the designer must rely on the user's capabilities tc make implications. To illustrate, suppcse a decision has been made to position a line number as the first argument in any command in which it is required. A command would be of the form '10d' which, in a text editor, might mean "delete the 10th line of text". In another mode it may be best to form commands to coincide with natural language. Such a command would be of the form 'delete <entity> <line number>', i.e., 'delete name 10" where name is deleted from line 10 [BARN81].

Consistency within a mode should be a high design priority. Distinct modes within a computer system may require different command structures. The increase in clarity resulting from different command structures must be weighed against the decrease in consistency. If a system contains more than one mode, each mode should be clearly identified. Commands should be unique tc each mode, thus reducing the potential of error if a command is used in an incorrect mode [NORM83]. These guidelines will allow the user to make correct implications regarding the appropriate commands for a given mode.

2.2.3. Experimentation

Human beings do not accept a list of rules once and for all and then never break these rules. People make mistakes, both deliberate and accidental, and then learn from their mistakes. "Errors in dialogue are not essentially key problems to be avoided at all costs. The naive user should feel free to make errors as part of their exploration of the system" [GAIN78]. If users are given the option of trial and error several prerequisites must exist:

1. Penalties for errors must not be excessive.

2. Cost in time, money and labor must be low enough to allow for several trials.

3. Results of each trial must indicate changes to be made in later trials.

4. Results of a trial should be given quickly.

These prerequisites require cheap and plentiful computing power which has not been available in the past. Because computer power has decreased in cost it is now reasonable to expect the user to experiment with his computer system. Effective computer dialogues will inform the user if his trial results in an irrecoverable state and will also give

him estimates to the cost of his requests, thus allowing the user to experiment intelligently.

2.2.4. Motivation

Although computers do not have feelings toward humans, humans definitely have feelings towards computers. "Few people who have interacted personally with a computer have lukewarm feelings about the experience. Most often people find the interaction either highly enjoyable or totally distasteful" [HEIN75]. Man-machine dialogue must be designed in such a way as to nurture a positive attitude toward the computer. The user has direct contact with his terminal and treats the remainder of the system as a black box. He is concerned that he obtain the appropriate responses to his requests in a reasonable amount of time. One way the user measures the quality of service is by the features found within his dialogue with the terminal and the ease with which his dialogue takes place [HEBD78]. The user's interaction with his terminal can be viewed as the highest level virtual machine in a hierarchy of virtual machines. It is critical that this virtual machine present itself as understandable and sympathetic to its users [GAIN78]. As a user communicates with a computer his confidence in it should increase. This can be done through respectful responses to his commands, a feeling that his data is safe because of

systematic, predictable and consistent dialogues and by being provided with pertinent information as to how his requests are progressing.

2.3. Design features found in user friendly interfaces

The previous discussion of underlying principles of human communication provide a basis for the discussion of specific principles which can be incorporated in a user friendly interface.

2.3.1. Mental models

> System designers and applied psychologists are increasingly coming to believe that people deal with complex interactive devices by making use of a conceptual model of the device" [YOUN81].

A user's conceptual model is a set of concepts the user assembles to explain the organization and behavior of his system. Often the model is a analogy to a real life activity or object. The Star Information System by Xerox incorporates into its system a strong analogy to the office environment with which its user is familiar. The display screen shows the common office furniture: a desk, mail box, file cabinet, etc. A user sees the result of his commands on the screen as functions are executed [SMIT82].

Research has shown that the development of mental models does affect learning and memory. In 1979 Mayer concluded that advance organizers (introductory materials presented in advance of learning) have an effect on the learning of technical material [MAYE79]. A user's model is a form of advance organizer and will have an effect on the user's relation to a computer system. In 1975 Mayer conducted an experiment where subjects learned a subset of the BASIC programming language. The subjects were divided into two groups. One group was given a model which made analogies between the four major functional units that are present in programming languages: a ticket window analogy for input, a note pad for output, an erasable score board for memory and a shopping list for executive control. The second group was not given a model. Both groups read a booklet that defined and gave illustrations of seven BASIC commands. The results of the study showed that the group possessing a mental model was better able to determine what tasks a given sequence of commands would perform [MAYE79].

If a user is not presented with a model of his system he will formulate one himself. The formation of the model may not be a conscious effort by the user. The human mind constantly searches for patterns of cause and effect. A danger exists when the user generates his own model. If

the user formulates a false or superstitious model his effectiveness will decrease. Designers can guide the user in formulating correct models by avoiding non-deterministic behavior within a system and by clearly showing the user the consequences of his actions.

A transparent system that is totally open and understandable will also contribute to a correct model. The user should know why the system is doing what it is doing. If a novice user has a broad understanding of how the components of his system relate to each other and an understanding of the overall system structure, he feels confident as he navigates throughout the system. In contrast, a user with knowledge limited to the specific commands needed to execute a given task will be fearful of the consequences, if he deviates from these commands [SELA82].

The user should feel the system is controllable and non-mysterious [ROBE81]. For example, when more than one level of dialogue is included in a system, if the user has control over the dialogue, the changes in interaction will not come as a surprise nor seem mysterious or unpredictable.

The designer may choose to present a model to the user or may choose to create an atmosphere in which the user forms a valid model independently. There have not

been extensive guidelines developed to assist designers in the invention of models. Further investigation is also needed to study the variance between models of the same system by different users [NICK81].

2.3.2. User expertise

"The design of many current computer systems is based on the assumption that there is a single ideal system which can satisfy all users, but it is doubtful that any single system is best for any and all users" [BLAC81]. A user friendly system must acknowledge the varying abilities of the users. Generally users can be divided into two groups: novice or infrequent users and expert or frequent users. What users expect from a system and the way a system is used will vary greatly between the two groups.

Flexibility within a system is required to meet the continuum of abilities among its users. Dialogue formats such as menu displays and form filling are easy to learn and provide rigid structure for the novice user. On the other hand, programming style dialogues provide rapid interaction which is desirable for the expert user. If a system will be used by both groups provisions must be made for each group. Hall describes a library retrieval system which incorporates both menus and a command language. There are two distinct paths for retrieving information.

The user is allowed to select his approach and is able to switch between the two paths at will. The novice user is not overwhelmed with a terse set of commands and the expert is not frustrated by being required to traverse a sequence of menus to accomplish a task [HALL78].

"A truly friendly system must afford both intimacy as well as distance" [WERN82]. A novice user should have interaction maximized. He should be presented brief messages and be required to make simple responses. A continuous interaction reinforces his ability to function successfully within the system. An expert user would prefer a quieter system that allows him to progress quickly through a task with a minimum amount of interruption.

The designer must step outside his own experience and view the system from the user's eyes, hence examining the system from the outside in [DWYE81b]. Consider the input statement in BASIC. A user can be presented with either of the following statements:

```
INPUT THE INTEREST RATE PER YEAR
?
```

```
WHAT IS THE INTEREST RATE PER YEAR?
```

A designer feels completely at ease with the first

interaction because he has a clear understanding of the word 'input'. The naive user may be confused with the word and unsure of what action is expected of him. Clearly the the second interaction relates more closely to a naive user's concept of data entry [HEIN75].

2.3.3. Mental load

Men and computers each have their own strengths and limitations. Man is skillful in selecting goals, observing patterns, and handling unforeseen circumstances; however, he is slow, forgetful and prone to error. In contrast the computer is extremely fast, accurate and does not forget, but it does not have man's reasoning ability. The designer must carefully balance the strengths of both man and machine. A computer can be used to complement the user by minimizing the mental load on the user. Because the computer's memory is better than man's, the designer should rely on the computer memory as much as possible [SIMP82]. Four methods for reducing mental load are described below.

Mental load is reduced in a system which makes use of default options where the user is given the most likely response to an interaction. The user is then free to concentrate on the exceptions he encounters. Defaults can also be incorporated into the operating system. The RSTS

operating system automatically makes a backup of each file when it is edited. The user is not required to create his own safeguard if he does irrecoverable damage to his file while editing. The user does not have to express his desire to have his editing changes written to the file at the end of the editing session. UNIX, however, requires the user to give a write command before he finishes editing. If the write command did not have to be explicitly given by the user, the editor would be friendlier to the naive user. It is likely that a new UNIX user will be frustrated at least once by neglecting to use this command.

A second technique for reducing the mental load is to provide prompts that are brief, specific, and show the desired entry format. If the user must enter a date, he should be shown the appropriate form of the date. He should not be left to guess the expected entry format.

The use of menus in man-machine dialogue reduces mental load. The user is not required to memorize special commands or names; rather he must only recognize the desired task from a list of options presented to him. Care must be taken to ensure only relevant options are presented to the user. Too many options or ambiguous options will confuse the user and increase his mental load.

Fourthly, there are applications that can be structured such that the user's response consists of only "yes" or 'no' to a series of questions. This type of interaction is effective in medical diagnosis systems where the user may have no computer experience.

It is not always expedient to eliminate user memorization. Research has shown that when humans have memorized information, they are better able to use that information to make inferences relating to it than when the information is available but not memorized. If a system is a tool which will be used consistently by the user over a period of time, the user may use the system more effectively if he has committed pertinent information to long term memory [BLAC81].

Mental load to the user will be increased when his system is non-uniform or inconsistent. If the dialogue between user and computer are similar throughout a system except for a few tasks, or if help is available for most tasks but not all, the system reduces its friendliness. As the user becomes familiar with the system, certain features he expects may not exist.

> This not only puts an additional memory load on the user but can also be devastating psychologically when a naive user has at last gained confidence in his familiarity with a system and then suddenly finds that his trust in himself and the system is misplaced [GAIN78].

2.3.4. Response time

System response time is a vital issue relating to interactive systems. Nagel presents the following guidelines for acceptable response time: [NAGF78].

> 15 sec	Intolerable
> 4 sec	Too long in most cases - Possibly tolerable after termination of major work step.
> 2 sec	Too long for high concentration work
< 2 sec	Necessary for work consisting of more than one step to be bridged mentally.
< .1 sec	Immediate reaction to key stroke

Long delays in system response are frustrating to the user and will reduce his efficiency. If delays exceed 15 seconds the system no longer provides conversation-like interactions. The length of delay is not the only consideration in determining acceptable response times. "A long delay may be more tolerable if one can predict when it will end than if one cannot" [NICK81]. A user friendly system should warn the user when a task will take a longer than the usual period of time to execute. For example, if a user requests an archived file, he should be given an estimate of the time needed to retrieve the file. In UNIX a program compilation will take longer than a simple request for the current date and time. The user will not

be alarmed by a delay if he expects one. Also if a user believes that his task is causing the delay, he will be more tolerant of the delay.

User expertise contributes to the definition of acceptable response time. A naive user requires time to carefully read dialogues and formulate the appropriate responses. He would be less likely to notice a delay in response time. In contrast, the expert user who is familiar with the system dialogues will be less tolerant of delays.

Response time will contribute significantly to the degree in which a user will experiment within a system. If there is a low response time a user will be more likely to explore and discover the full potential of his system. The Zog man-machine interface was designed to provide the user 'instantaneous' results. Zog is a menu driven interface to large networks of data. The user is presented with a new frame instantly upon requesting an option within a menu. He can easily scan a new menu frame and not be penalized with a lengthy time delay [ROBF81].

2.3.5. User feedback

Closely related to response time is the concept of user feedback. "It is disastrous to the user's model when he invokes an action and the system does nothing in

response' [SMITE2]. If the user does not see a response to his commands he is likely to repeat his previous actions because he is not sure whether the system has heard him. Such interaction could lead to undesirable results.

A user should be sure of his orientation within a system. The user will be more confident when he knows where he is in relation to the total system as he progresses in his work. "If the user internalizes the system's structure, he will feel more comfortable working with the system' [SELA82]. The tcp level menu provides a sense of direction and the ability to reach a place of stability within a menu driven system.

The system feedback a user receives can be used to modify his behavior. If the user is presented with positive phrases acknowledging his responses, the user is likely to repeat his correct actions. Words such as "OK" and 'GOOD' can be used as rewards. Such rewards are most effective when they are presented immediately. On the other hand, computer systems which only acknowledge incorrect actions will increase the user's sense of failure, thus decreasing his effectiveness. System feedback should not be harsh or demeaning [DWYE81a], [DWYF81b].

2.3.6. Error handling

Error messages reveal characteristics of a system. Messages such as 'FATAL ERROR, RUN ABORTED' confuse and frustrate users. A novice user may quickly develop a negative image of his computer system if the error messages are hostile or obscure [SHNE82]. A user friendly system presents the user with polite and factual error messages. These messages should not be designed to humor, punish, or intimidate the user.

Error handling is a three step process. First the user must be told an error exists. A designer may choose to alert the user by sounding a beep or blinking the screen. These techniques are only effective if they are not an integral part of the normal system activity. It is also effective to reserve an area of the screen for error messages. When text appears in the area the user is alerted to an error. Secondly, the system must identify the errors. When describing error it is best to use non-computer jargon. The designer should select words that are understood by the user. The error message should be as specific as possible. If a user enters a letter when a number is expected he should be told that a letter is not an appropriate response instead of the vague message - 'invalid entry'. Lastly the user should be directed to the correct response for the error situation. In the

previous example the user should be told that only numbers are acceptable entries [HEIN75].

Galitz presents the following techniques which lead to easy, correct and fast message interpretation [GALI81].

Do not use abbreviations
Do not use contractions or short forms
Use brief, simple sentences
Use affirmative statements
Use the active voice
Order words chronologically

Again, the expertise of the user dictates how explicit error messages should be. The naive user needs lengthy error messages whereas a cryptic reminder may be sufficient for the experienced user. It would be desirable for a system to contain a multilevel approach to error messages. The user should control the amount of detail he is given on an error condition.

Extensive error checking at data entry will minimize the number of errors within a system. Data syntax should definitely be checked; however, care must be used when rejecting data which deviates too far from the norm. A designer may not be able to predict all acceptable responses but he is able to warn users if values are

suspicious.

Tasks which result in major changes to a system should require revalidation of the task requested. In a database system a user should be informed of the consequences when he requests deletion or purging of a file. He should be asked to verify his request, thus safeguarding himself against costly errors.

A further extension pertaining to errors is a system which identifies errors and then corrects them without user interaction. Such a system would be difficult to implement, for often times it would be difficult to determine a user's intention. The following two examples illustrate situations where error correction would be desirable. Naive users may know what they want to do but are unsure of the appropriate commands. If they are able to remember part of a command the system could supply the remainder. A 'C' programmer can easily forget to include a semicolon at the end of a line. If the system inserted a semicolon as a default the programmer would be relieved of another detail.

Error handling is costly in terms of system design and implementation. Layered error messages and error correction increase run time overhead. The system designer must evaluate the optimal balance between cost

and benefit.

Frror handling is one of the most important aspects of the man-machine interface, yet it is often neglected. Designers must give error messages a high priority. A well thought-out system of error messages coupled with improved displays will increase the quality of a user interface [BROW83].

2.3.7. Help facilities

All users of a system need assistance in using their system. The amount and type of help required is dependent on the level of user expertise. A novice user is anxious to do something useful to him as quickly as possible. "Intrcductory training material should be designed to bring the beginning user to the point of accomplishing something of interest quickly" [NICK81]. The new user does not need to know complex details which show him how to use the full power of his system. As his level of expertise increases further training and dccumentatien with greater detail become significant and interesting to the user. Once a groundwork of fundamental capabilities is mastered the user is in a position to discover and learn advanced tasks.

It is desirable to have an on-line help facility. A large manual is formidable to both the new and experienced

user. When the user has a question he appreciates an answer available on demand placed at his screen. An excessive amount of material on a screen, however, can distract the user and discourage him from reading the material. A layered approach within a help system would be effective in relieving the information overload. The top layer in the help system should present short messages which briefly list appropriate responses. If the user desires more detailed help he can continue to ask for help, thus being lead through additional layers within the help structure [GAIN78]. The top layer of help system should be extremely sensitive to the new user in its presentation. It is most frustrating to require help to understand the help message.

A system which logs user responses can be integrated into the help subsystem. The records can be analyzed to see the system capabilities the user has employed, the types and frequency of errors, as well as the general flow of dialogue. A supervisor could use the log to encourage users and offer guidance. A more sophisticated system could make use of a log to initiate suggestions to the user. A user may lack the knowledge required to even ask for information. A user is unlikely to explore and search for a more efficient method to complete a task once he is comfortable with an established method. A log could be

utilized in system suggestions to the user for alternate methods.

The existence of on-line help systems does not eliminate the need for hard copy documentation. Some information, e.g. diagrams, is presented more effectively in hard copy. A well organized user's manual is often easier to read than text presented at a terminal. Long descriptions to which the user must frequently refer would be valuable in hard copy.

2.3.8. Screen design

The screen layout affects the user effectiveness and enjoyment within a system. "As a general rule, access screens by paging, not by scrolling. People find it easier to read stationary pages than moving pages" [SIMP82]. A designer should use his capabilities to clear the screen when moving from frame to frame within a system.

Galitz offers several general guidelines for effective screen design. Screens should be presented in an orderly manner. The mind quickly identifies a cluttered screen and will either consciously or unconsciously exert effort by mentally reorganizing it into a meaningful form. A screen density of about 15% is easy to discern as compared to hard copy text that allows for a 40% density.

The eye is drawn to the upper left corner of the screen, thus the designer should use the upper left corner as the starting position. The designer should center titles and attempt to balance information on either side of the vertical axis. The screen can be divided into sections through the use of color, reverse images, lines, etc. The screen, however, should not be divided into too many small windows which gives the screen a cluttered appearance. If both upper and lower case are available on the terminal, both should be utilized. Lower case is less forbidding than upper case. Upper case is effective for labels and captions as it reduces search time. Attention getters such as blinking, reverse video, higher intensity and changes in character size must be used in moderation to be effective. Overuse of attention getters can be distracting.

2.4. Cost considerations

The design of a user friendly interface is an ongoing process. As users interact with a system they will express comments, complaints and suggestions. The system designer will eventually break his ties with a system. Before he does, he will probably be asked to make changes in the man-machine interface. Careful preliminary design will make changes easier to incorporate. The system design will contribute to the ease of maintenance as its

use continues.

A user friendly system will take more time to design and implement but this should not be an excuse for less than optimal interactive systems. Project milestones and financial constraints may force a designer to allocate insufficient time to study the user's point of view. Although there is an immediate savings in cost, the savings may result in long term dissatisfaction.

> We must not loose sight of the end objective of the programming activity; to provide an effective implementation of the user's requirements. The conflict arises not because programmers do not want to achieve that objective, but features which make marginal improvements to the usability of a dialogue may appear to require a disproportionate amount of development effort. The problem then becomes one of how you measure the benefit feature and financial terms [HEED78].

As computers are increasingly utilized in the work force the ease with which they interact will directly affect job satisfaction. Consequently, a user friendly system will have an effect on the financial concerns of the institutions in which they are incorporated.

CHAPTER 3

DESIGN AND IMPLEMENTATION OF USER FRIENDLY INTERFACE TO UNIX

3. Design and implementation of user friendly interface to UNIX

3.1. Design goals

3.1.1. Primary objectives

The user friendly interface to UNIX was written with two primary objectives. First the interface must be easy to use. An individual should be able to interact with the UNIX operating system without prior UNIX instruction. Secondly, the interface must provide the user an opportunity to learn UNIX commands. The first objective of providing an environment where the user can function effectively could be accomplished without informing the user of the corresponding UNIX commands. However, if the user is notified of the appropriate UNIX commands for each task he requests, a by-product of the man-machine interaction will be the user learning UNIX commands. The training aspect of the interface is important to the new user who will continue to use UNIX for an extended period of time.

3.1.2. Secondary objectives

A secondary design objective addresses the flexibility of the interface. The process of selecting the type of dialogue and the implementation of that dialogue included the consideration of adapting the interface to other applications. It was desirable to design an interface that could be easily modified and be incorporated into other systems where user friendliness is a priority.

The size of the interface program was a significant design consideration. It was desirable to keep the interface as small as possible. It was projected that the interface would be used by many users simultaneously. A small interface would place less demand on the computer system, increasing overall system response time.

3.2. Audience using the interface

A major design decision addressed the audience using the interface to UNIX. Three classes of potential interface users were established:

1) New UNIX users, in particular beginning programming students

2) Experienced UNIX users who have been away from UNIX for an extended period of time

3) Casual users, individuals who do not use UNIX regularly.

The primary audience has no UNIX experience. This class can be divided into two subclasses. One subclass consists of individuals who have prior computer experience. These users have an understanding of the tasks they want completed, but do not know how to have them performed in the UNIX environment.

The second subclass has no computer experience. These users require support in developing a model for a computer system. They need help in understanding fundamental concepts such as what is a file, what does it mean to edit, what is a directory, etc. The interface written for this project is not a tutorial on computer systems. It does not attempt to define or provide a model for underlying concepts. Instead, it does provide a means of functioning effectively within the UNIX environment. This interface is beneficial to the beginning programming student in that his anxiety related to learning a computer system is reduced; thus, the beginning student is able to appropriate more time and effort toward developing programming skills. Classroom experience will provide the underlying concepts and models.

The second class of users consists of experienced UNIX users who have been away from UNIX for an extended period of time. Such a user has a series of tasks to be completed but is unsure of the syntax of the appropriate UNIX commands. This user will benefit from seeing the UNIX commands as his tasks are performed. The user will accomplish his tasks and at the same time be reminded of correct UNIX commands. An experienced user will spend a brief time within the interface. A quick reminder will enable him to use UNIX effectively once again.

The third class of users includes individuals whose experience with UNIX is limited. These users may occasionally want to use UNIX but will not use it frequently enough to warrant their learning the system. This class of users will not be particularly interested in the training aspect of the interface. They will, however, appreciate the ease with which their tasks are completed.

3.3. Length of time spent using the interface

The interface is not designed for long term continuous use. As an individual becomes familiar with UNIX he will appreciate its terseness. An interface between the user and UNIX would become a nuisance over an extended period of time. The interface should, however, provide for the transition period when the user needs the security

of the interface while at the same time would like the freedom to initiate a subset of commands on his own.

3.4. Scope of the interface

The scope of the interface is limited to common UNIX functions (see figure 1). The tasks included reflect the activities a beginning programming student uses. Once an individual has spent time using the UNIX interface, he will be better equipped to understand UNIX documentation found in manuals. The user will have established a foundation of common commands and their usage. He will then be able to build on this foundation independently. The user will be able to compare and relate less frequently used functions to the ones with which he is familiar.

3.5. Model for interface

The TENIX 8560 Multi-User Software Development system by Tektronics includes a program named Guide. Guide helps the user learn about the TENIX operating system while he uses the system. Guide incorporates a menu structure. The TENIX system provided a model for the interface implemented by this thesis project [TEKT81].

3.6. Dialogue selection

The dialogue requirements within the interface can be classified into two types. First, the user must

---

1) Introduction to menu shell

2) Introduction to terminal

3) Change level of prompt

4) File manipulation
    1) Return to top level menu
    2) Create a file
    3) Edit a file
        1) Display contents of file
        2) Display a screen of text beginning with a given line
        3) Display a given line
        4) Append after a given line
        5) Insert before a given line
        6) Delete a given line
        7) Exit the editor
    4) Copy a file
    5) Move a file (change its name)
    6) Delete a file
    7) Display a file at terminal
        1) Return to previous level
        2) Scrolled display of a file
        3) Text displayed one screen at a time
        4) Octal display of a file
        5) Display non-printing characters in a file
        6) Temporarily exit menu shell
    8) Display a file at printer
    9) List names of files
    10) Temporarily exit menu shell

5) Directory manipulation
    1) Return to top level
    2) Create a directory
    3) Delete a directory
    4) List contents of a directory
    5) Change directory
    6) Path identification
    7) Temporarily exit menu shell

6) Other system functions
    1) Return to top level
    2) Current date and time
    3) Who is working on the system
    4) Send mail
    5) Receive mail
    6) Write to another user

```
        7)  Display UNIX manual
        8)  Change password
        9)  Temporarily exit menu shell

7)  Program compilation
        1)  Return to top level
        2)  Pascal program
        3)  C program
        4)  FORTRAN program
        5)  Temporarily exit menu shell

8)  Temporarily exit menu shell

9)  Exit
```

Figure 1. Scope of menu shell

---

communicate the specific task he wishes to have completed. Secondly, the user must supply the pertinent information that a particular command demands. A combination of menu selection and question-answer dialogues was selected to accommodate the communication flow between the user and UNIX. A series of menus satisfies the first class of communication. The user is lead through a sequence of menus resulting in the selection of the task to be completed. Once a task is selected, the second type of communication is satisfied with a question-answer dialogue. The question-answer technique was selected due to the diverse nature of tasks available to the user within the UNIX interface. The information required for compiling a program differs from the information required for sending mail, printing a file, changing directory, etc. Question-answer dialogue provides the flexibility needed

to gather the broad range of responses required for the available tasks.

The literature supports the ease with which a menu system is used. Menu dialogues are the archetype of computer-initiated dialogue [RAMS79]. A menu system presents the user with a list of alternatives. The user then selects one of these alternatives. A menu system is especially helpful to the new user in that it provides a list of available options. Many times the new user may not have sufficient knowledge to understand what are and what are not appropriate requests. The computer is responsible for the majority of the man-computer interaction, thus the user's participation in the dialogue is minimal. The passive role of the user contributes to the rapid speed in which a menu system can be learned.

Menu dialogues have a major drawback. Once a user is familiar with a system, the menus which once provided a pleasant environment become a nuisance. For this project, the cumbersome characteristic of a menu system becomes a positive feature. The objective to teach UNIX commands is enhanced because the user is motivated to learn UNIX commands and thus be able to work independently of the menu structure. The user realizes he is able to initiate a command on his own faster than he is able to using the menu interface. The frequent UNIX user is encouraged to

become self-sufficient.

3.7. Implementation of the interface

A menu dialogue naturally lends itself to a hierarchical structure. The first menu presented to a user corresponds to the root in a tree structure. As the user makes selections he traverses the tree. When the user reaches a leaf node, he is prompted for the information related to his task, and finally his task is executed.

3.7.1. Language selected The program driving the menu structure is written in "C". The C language provides a mechanism that allows one process to create another process, wait until the new process completes execution, and then continue in its own execution. The ability to 'fork' a child process is particularly suited to this project. Most of the UNIX operating system is written in C. The close ties between UNIX and C also point toward C as the language for the interface.

The leaf node programs are written as shell scripts. Shell scripts are used because of their error checking capabilities. Increased error checking within the shell script programs prevents the user from receiving obscure error messages from UNIX. The shell script programs also increase the flexibility of the interface. If a question-answer dialogue must be changed, the file

containing the shell program can be modified without recompiling the entire interface.

There are two exceptions to the use of shell scripts at the leaf nodes. The option which edits a file required a program which was able to spawn a child process. The edit option was written in C. Thus, the process was able to fork the UNIX editor as a child process. The edit program simulates the menu selection procedure found in the driving program.

The second exception was the change directory option. In the C language when a child process returns control to the parent process, the child's environment is replaced with the parent's environment. A shell script program could be written to change directories; however, that directory change would be nullified when the process completed. When control returned to the parent, the child's environment would be lost and the current directory would remain the parent's current directory. When a user requests the change directory option, a simple shell script program is executed. The single function of the shell script is to exit with a status code. This exit status signals the driving program to call a function which performs a directory change. This function follows the dialogue format found in the shell script programs. Because the directory change is executed within the

driving program, the new directory remains the current directory until another change of directory is requested.

3.7.2. Structure of the interface The program driving the menu structure is short and straightforward. Figure 2 describes the flow of the driving program.

Each option within a menu display is associated with a separate file. As the user makes a selection from a menu display, the driving program opens the file associated with that option. If the file is another menu display file, the driving program repeats the cycle of menu display and registering user response. If the file

---

1) A brief introduction welcoming the user and describing the appropriate response to a menu display.

2) Repeat until "exit menu" is requested

   a) Display menu

   b) Register response
      If a leaf node is selected
      then
         execute the file associated with leaf node
         return to previous menu
      else
         retrieve associated menu file

Figure 2. Program flow of driving program

---

is a leaf node, the program named within the file is executed. At the completion of a leaf node task, the user is presented with the most recent menu.

The path taken within the menu structure is recorded in an array representing a stack. The array consists of pointers to the names of the files accessed in the path from the root (top level menu) to the current position in the menu structure. The stack is used to determine the return path.

The menu display system consists of 3 major activities.

1) Initialization
2) Option menu processing
3) Leaf node processing

The initialization activity presents the introductory statements to the user. The stack pointer to the root or top level menu is also initialized (see figure 3).

The option processing activity navigates a series of menus until a leaf node is reached. The user must respond with a valid response to the menu options. He is instructed as to the appropriate responses if he gives an invalid response. This activity also maintains the stack array representing the path taken (see figure 4).

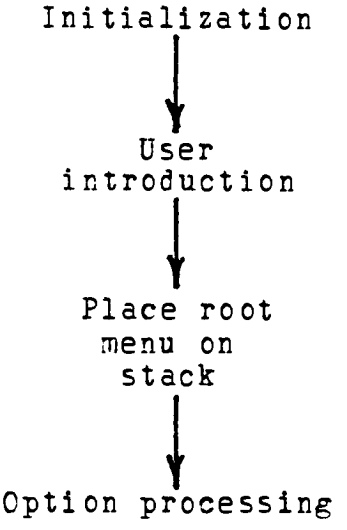

Figure 3. Initialization activity

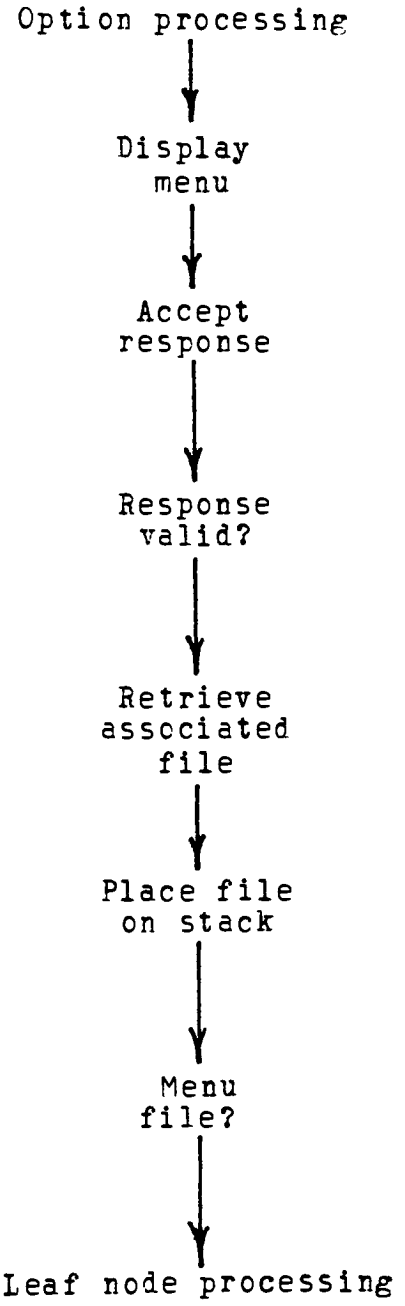

Figure 4. Option processing activity

The leaf node activity initiates the request to fork the executable file associated with the menu option selected. The exit code from the child process is evaluated and the appropriate action is taken (see figure 5).

3.5. Description of files The files accessed by the driving program are in two forms. A menu display file is shown in figure 6. A menu display file consists of the number of available options in the menu, the menu title, the list of menu options and the file associated with each option.

The form of a leaf node is shown in figure 7. A leaf file uses the exclamation mark to distinguish it as a leaf file. 'File name' represents an executable file which prompts the user for pertinent information and initiates the appropriate UNIX commands.

Communication between executable files and the driving program is accomplished with the exit status of the executable programs. Most programs exit normally with a status of 0. The user is presented with the most recent menu and option processing continues. The following non-zero exit status have special meanings.

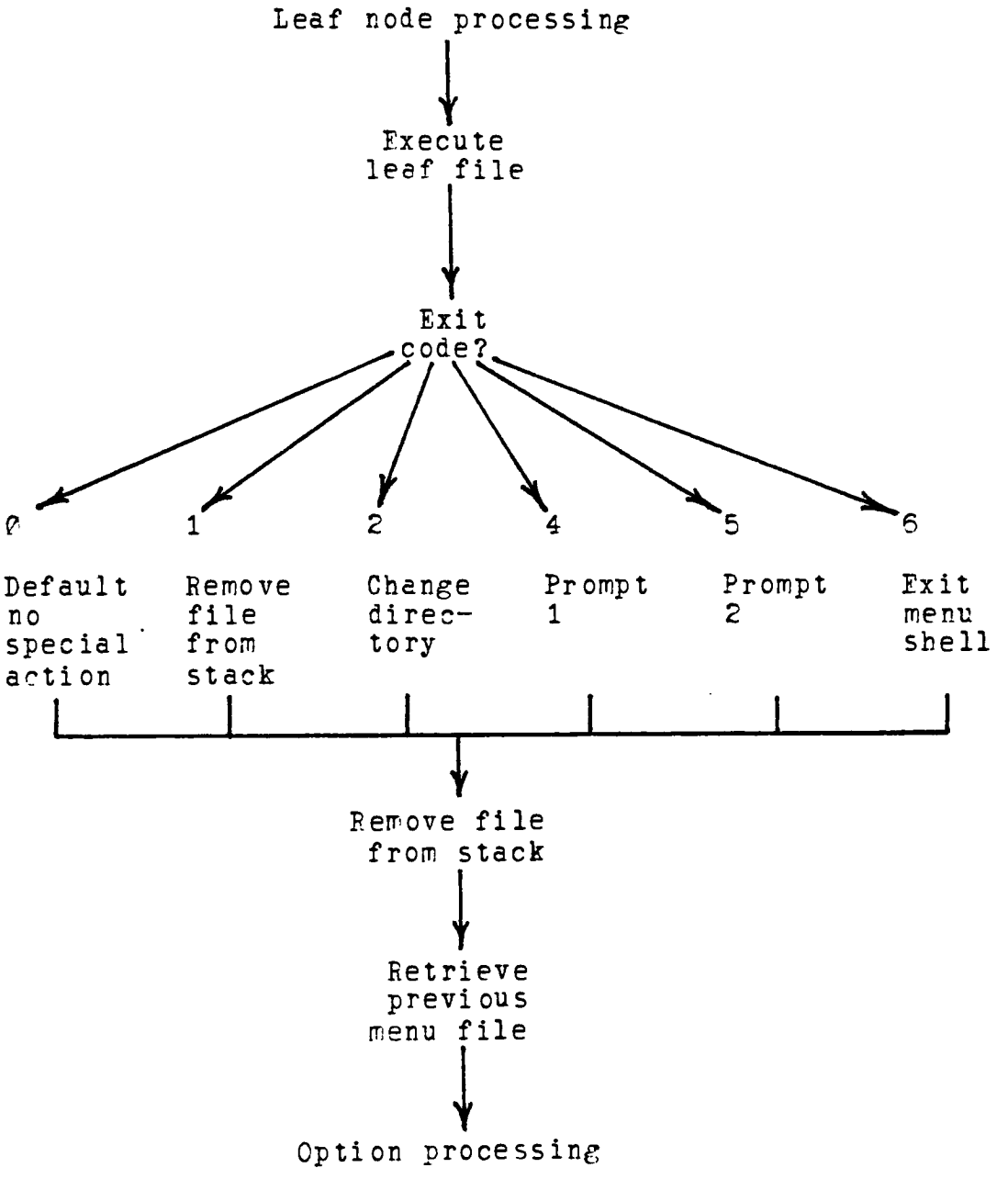

Figure 5. Leaf node processing activity

```
#n

                    MENU TITLE

      1)  Menu option 1
+file name associated with menu option 1
      2)  Menu option 2
+file name associated with menu option 2
      .
      .
      .
      n)  Menu option n
+file name associated with menu option n
```

Figure 6. Example of menu display file

```
!file name
```

Figure 7. Example of leaf node file

1. Change directory
2. Return to previous level menu
4. Change to verbose prompt
5. Change to terse prompt
6. Exit menu interface

A status code of 3 does not have a special meaning. If a

user breaks out of a shell script, the exit status is 3. In this case the user is presented with the most recent menu and option processing continues, thus no special action is required.

3.9. Significant features of the interface

The interface is flexible. In this application the driving program initiates execution of shell script programs. The only restriction the driving program places on the leaf files is that the files name executable programs. The interface can easily be modified for other applications. Appropriate menu options must be specified in menu files and the corresponding executable programs must exist; however, the executable programs may be written in any language and may use the dialogue technique most appropriate to the new application.

The interface provides a means for the user to learn UNIX commands. Each time a program initiates a UNIX command, the interface displays the command at the terminal. Each command is preceded by three exclamation marks, thus, the user is able to recognize UNIX commands from other dialogue. For example, if a user requests the "current date and time" option, he is presented with the following command immediately before the date and time are displayed.

!!! date

Many UNIX commands require arguments. There are two levels of prompting users for these arguments. Level 1 is a verbose prompt. The user is presented with complete questions which reinforce the nature of the task requested. Level 2 is a terse prompt; user prompts are limited to key phrases. For example, if a user requests the "delete a file" option, the verbose prompt asks:

What is the name of the file you wish to delete?

The terse prompt simply asks:

File name?

The interface initially presents the verbose level of prompt. A menu option allows the user to change the level of prompt from verbose to terse and vice versa. When a user first interacts with the UNIX interface, the verbose prompt is appropriate. Once a user is familiar with the tasks available, the terse prompt will increase the speed of interaction between the user and the interface.

The user is in control of the level of prompt. The literature supports the concept of user control [ROBE81]. If the user makes the decision to change the prompt level, his sense of control over the system is increased. If the

prompt level changes without explanation or user request, the system will seem confusing or mysterious. The user will formulate a reason for the change in dialogue. His conclusions may not be accurate and his confidence in the system may be damaged by an inaccurate model.

The leaf programs include a limited help facility. The user is asked questions pertinent to the task he has selected. If a question is preceded with an "*", the user has the option of entering a question mark as his response. A question mark results in the presentation of additional comments regarding the question the user was asked and the nature of the task requested. The responses to the question mark option are written into the executable programs. The new user will find the additional comments helpful. He would be wise to make use of the question mark option. However, as the user becomes familiar with the commands, he will not require this support. The additional explanations are not part of the basic dialogue. The help dialogue would reduce user efficiency once the user becomes familiar with a given task. The help messages are readily available; however, the user must request them.

Most users will use the interface on a temporary basis. A user will quickly learn a subset of UNIX commands and will want to try these commands independently.

At the same time, he will want the security of the menu interface for those commands of which he is unsure. The interface provides for the transition period by including a temporary exit option in each menu. The temporary exit option accepts the UNIX commands a user requests and executes these commands. The interface does no error checking in relation to the commands, and the user sees the normal UNIX response to his commands. The user is able to return to the menu dialogue at any time. This option allows the user to experiment on his own. He is not threatened by the syntax errors he makes because he realizes he is able to complete his task by returning to the menu structure if he is unsuccessful on his own. When the user enters the temporary exit option, he must take responsibility for his commands. For example, if the user wishes to copy a file and reverses the input and output arguments, his task will be completed and he must accept the consequences of his command. Because the temporary exit option is included in each menu, it is readily available at any time. It is easy to request the option and easy to return to the menu structure, thus the user is encouraged to initiate UNIX commands independently.

Execution time varies between UNIX commands. Program compilation and execution require significantly more time to complete than other UNIX commands such as file display

and listing the contents of a directory. The interface includes a warning message to the user when a task will have longer that usual execution time. These warning messages will reduce user anxiety. The user will expect a delay in response and will be more tolerant of the delay.

The interface dialogue is straightforward and polite. Complete sentences are used whenever messages are printed. The words and phrases used in messages are meaningful to a new user; however, UNIX terminology was not avoided. Most users will eventually interact with UNIX independently; they must become acquainted with UNIX terminology.

Both upper and lower case characters are used in the interface. Menu displays use upper case characters to list available menu options. Leaf programs primarily use lower case characters which coincide with normal English text. All menus and leaf program have titles. The titles help the user verify his current location within the menu structure.

The above features illustrate how many of the characteristics of a user friendly interface were incorporated into this interface. The interface is easy to use; however, each time a user makes a response, he must press the 'return' key. Other techniques such as cursor movement and cursor position could have been used. Such techniques

would initially be easier for the user; however, they do not prepare the user for future UNIX terminal interaction.

CHAPTER 4

EXTENSIONS

4. Extensions

This thesis project accomplished the goal of providing an interface to UNIX enabling an inexperienced UNIX user to function comfortably within the UNIX environment while at same time teaching a subset of UNIX commands to the user. There are improvements to be made which would increase the quality of user friendliness and would increase the scope of the project. Extensions to the project are discussed below.

The interface was written for use on non-graphic terminals. The user is responsible for knowing his position within the menu structure. This is not difficult in the present application as the menu structure is only four levels deep. A graphics display window showing the current position in the menu structure in relation to the entire structure would increase user orientation [SEIA82]. The issue of user orientation becomes more critical as the depth of the menu structure increases. As the number of levels increases in a hierarchical structure, the cost in time of returning to the top level increases. A visual display of the path taken within the menu structure would

eliminate the need for a user returning to the top level menu as a means of restoring orientation.

An option that returns control directly to the top level menu also becomes important as the depth of the menu structure increases. This option would be valuable on each menu display. The user would no longer be required to retrace his path, one level at a time in order to return to the top level menu.

The help facility available within the interface is limited. The leaf programs provide brief help messages related to the questions the users must answer. There is also a menu option which provides access to the UNIX manual command. However, the interface does not provide a comprehensive help system which is tutorial in nature. A help option could be incorporated into each menu display. Such a help facility should give an general description of the options available within that menu along with an explanation of underlying concepts related to the menu options. An extended help facility would assist the user in formulating a correct model of the system.

The interface includes an edit a file option. The editor is consistent with the menu design in that it is a menu driven editor. The edit option contains line oriented edit commands. It would be desirable to expand

the editing capabilities to include many of the character oriented edit commands. A screen oriented editor would also increase the friendliness of the edit option.

The temporary exit option within the interface provides a means for the user to work independently. When the user returns to the menu structure, he returns to his position prior to his temporary exit. The interface could be enhanced by paralleling the user's position within the menu structure while he works independently. The user would then return to the position within the menu structure associated with his most recent independent action. A parallel system of menu structure and command language would increase the mobility of the user and provide a mechanism for user short cuts. Hall describes a library administrative example which incorporates a dual interface of menu selection and command language and allows the user to move freely between the two techniques [HALL78]. A dual approach is effective in an environment which serves a varied population comprised of novice and expert users.

The subset of UNIX commands selected for this project was oriented towards a beginning programming student. Additional subsets of commands could be selected for various categories of users. The interface could be modified to implement these new subsets of commands. A subset could be written for an office environment. This

interface would place a greater emphasis on commands such as electronic mail, text formatting, and file manipulation. Another subset could be written exclusively for program compilation, debugging and execution.

CHAPTER 5

SAMPLE INTERACTION

5. Sample Interaction

Significant features of the menu shell are illustrated in the following sample interaction. Tasks have been selected from each of the top menu options with the exception of the "introduction to menu shell" and "introduction to terminal" options. The tasks included in the sample interaction demonstrate a user's movement within the shell structure as well as the dialogue he encounters.

WELCOME TO THE MENU SHELL

A menu of available options will be displayed at your terminal. You must type the number which precedes your choice of the menu options and press the return key.

TOP LEVEL MENU

1) INTRODUCTION TO MENU SHELL

2) INTRODUCTION TO TERMINAL

3) CHANGE LEVEL OF PROMPT

4) FILE MANIPULATION

5) DIRECTORY MANIPULATION

6) OTHER SYSTEM FUNCTIONS

7) COMPILE AND EXECUTE PROGRAMS

8) TEMPORARILY EXIT MENU SHELL

9) EXIT

Enter your choice of the menu options: 4

FILE MANIPULATION MENU

1) PETUPN TO TOP LEVEL MENU

2) CREATE A FILE

3) EDIT A FILE

4) COPY A FILE

5) MOVE A FILE (CHANGE ITS NAME)

6) DELETE A FILE

7) DISPLAY A FILE AT TERMINAL

8) PRINT A FILE AT PRINTER

9) LIST NAMES OF FILES

10) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 2

CREATE A FILE

*What is the name of the file you wish to create? ?

A file name is limited to 14 characters. Most characters may be used in a file name, however until you are familiar with the special characters and their meanings you should use only letters, numbers and the period.

*What is the name of the file you wish to create? sample

After seeing the UNIX command, enter the contents of your file. You are able to make corrections on the current line you are typing, however once you press the return key for a given line, corrections must be done at a later time by choosing the "EDIT A FILE" menu option.

TO INDICATE THAT YOU HAVE FINISHED ENTERING YOUR FILE PRESS THE "CONTROL" KEY AND THE 'D' KEY SIMULTANEOUSLY.

!!! cat > sample

This is a sample file. Its purpose is to demonstrate the "CREATE A FILE" option in the menu shell.

Press return to continue

FILE MANIPULATION MENU

1) RETURN TO TOP LEVEL MENU
2) CREATE A FILE
3) EDIT A FILE
4) COPY A FILE
5) MOVE A FILE (CHANGE ITS NAME)
6) DELETE A FILE
7) DISPLAY A FILE AT TERMINAL
8) PRINT A FILE AT PRINTER
9) LIST NAMES OF FILES
10) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 9

LIST NAMES OF FILES

```
*Do you wish a brief or long listing (b or l)?  ?

     A brief listing will only give the names of the files whereas a long
listing gives the mode, number of links, owner, size in bytes and time of
last modification in addition to the file names.

*Do you wish a brief or long listing (b or l)?  g

You must enter "b" for a brief listing or "l" for a long listing
(b or l)?  b


!!! l

sample              test.c              seesample

Press return to continue
```

FIIF MANIPUIATION MFNU

1) RETURN TO TOP LFVEL MENU

2) CRFATE A FILE

3) FDIT A FILE

4) COPY A FILE

5) MOVF A FILE (CHANGE ITS NAMF)

6) DFLETE A FILE

7) DISPLAY A FILE AT TERMINAL

8) PRINT A FILF AT PRINTFR

9) LIST NAMFS OF FILES

10) TEMPORARILY FXIT MENU SHELL

Enter your choice of the menu options: 7

DISPLAY A FILE AT TERMINAL MENU

1) RETURN TO PREVIOUS LEVEL

2) SCROLLED DISPLAY OF A FILE

3) TEXT DISPLAYED ONE SCREEN AT A TIME

4) OCTAL DISPLAY OF A FILE

5) DISPLAY NON-PRINTING CHARACTERS IN A FILE

6) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 5

DISPLAY NON-PRINTING CHARACTERS IN A FILE

Your file will be displayed with the non-printing characters in a readable format. The following formats are used.

^I = tab
^? = delete
$ = end of line

*What is the name of the file you wish to display? samplesee

File samplesee does not exist!

DISPLAY A FILE AT TERMINAL MENU

1) RETURN TO PREVIOUS LEVEL

2) SCROLLED DISPLAY OF A FILE

3) TEXT DISPLAYED ONE SCREEN AT A TIME

4) OCTAL DISPLAY OF A FILE

5) DISPLAY NON-PRINTING CHARACTERS IN A FILE

6) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 5

DISPLAY NON-PRINTING CHARACTERS IN A FILE

```
Your file will be displayed with the non-printing characters in a
readable format.  The following formats are used:
     ^I = tab
     ^? = delete
     $  = end of line


*What is the name of the file you wish to display?  ?

     You must enter the name of an existing file.

*What is the name of the file you wish to display?  seesample


!!! see -v -e -t seesample

^IThis file demonstrates$
the tab non-printing character$
and the end of line non-printing$
character.$


Press return to continue
```

DISPLAY A FILE AT TERMINAL MENU

1) RETURN TO PREVIOUS LEVEL
2) SCROLLED DISPLAY OF A FILE
3) TEXT DISPLAYED ONE SCREEN AT A TIME
4) OCTAL DISPLAY OF A FILE
5) DISPLAY NON-PRINTING CHARACTERS IN A FILE
6) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 1

FILE MANIPULATION MENU

1) RETURN TO TOP LEVEL MENU

2) CREATE A FILE

3) EDIT A FILE

4) COPY A FILE

5) MOVE A FILE (CHANGE ITS NAME)

6) DELETE A FILE

7) DISPLAY A FILE AT TERMINAL

8) PRINT A FILE AT PRINTER

9) LIST NAMES OF FILES

10) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 1

TOP LEVEL MENU

1) INTRODUCTION TO MENU SHELL

2) INTRODUCTION TO TERMINAL

3) CHANGE LEVEL OF PROMPT

4) FILE MANIPULATION

5) DIRECTORY MANIPULATION

6) OTHER SYSTEM FUNCTIONS

7) COMPILE AND EXECUTE PROGRAMS

8) TEMPORARILY EXIT MENU SHELL

9) EXIT

Enter your choice of the menu options: 6

OTHER SYSTEM FUNCTIONS

1) RETURN TO TOP LEVEL MENU
2) CURRENT DATE AND TIME
3) WHO IS WORKING ON THE SYSTEM
4) SEND MAIL
5) RECEIVE MAIL
6) WRITE TO ANOTHER USER
7) DISPLAY UNIX MANUAL
8) CHANGE PASSWORD
9) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 3

WHO IS WORKING ON THE SYSTEM

The following command lists the login name, terminal name and login time for each user currently on the system.

```
!!! who


rxf8947  tty02   Jun 10 14:12
rkd1649  tty07   Jun 10 15:12
rwf6565  tty09   Jun 10 15:09
jms2890  tty20   Jun 10 15:06
mkh1141  tty34   Jun 10 12:43


Press return to continue
```

OTHER SYSTEM FUNCTIONS

1) RETURN TO TOP LEVEL MENU
2) CURRENT DATE AND TIME
3) WHO IS WORKING ON THE SYSTEM
4) SEND MAIL
5) RECEIVE MAIL
6) WRITE TO ANOTHER USER
7) DISPLAY UNIX MANUAL
8) CHANGE PASSWORD
9) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 4

SEND MAIL

*To whom do you wish to send mail? ?

You must enter the login name of the person to whom you wish to send mail.

*To whom do you wish to send mail? mkh1141

*Is the message you wish to send contained in a file (y or n)? ?

You are able to send mail in two ways:
1) Create a file whose contents is the message you wish to send.
2) Enter the message to be sent at the terminal after the mail command is initiated.

*Is the message you wish to send contained in a file (y or n)? t

You must enter "y" for yes and "n" for no.

*Is the message you wish to send contained in a file (y or n)? n

After you see the UNIX command enter your message one line at a time. When you have finished entering your message press the control and "d" keys simultaneously.

!!! mail mkh1141

Hello mkh1141
This is a small message to demonstrate the send mail option.
Good bye

Press return to continue

OTHER SYSTEM FUNCTIONS

1) RETURN TO TOP LEVEL MENU
2) CURRENT DATE AND TIME
3) WHO IS WORKING ON THE SYSTEM
4) SEND MAIL
5) RECEIVE MAIL
6) WRITE TO ANOTHER USER
7) DISPLAY UNIX MANUAL
8) CHANGE PASSWORD
9) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 1

TOP LEVEL MENU

1) INTRODUCTION TO MENU SHELL
2) INTRODUCTION TO TERMINAL
3) CHANGE LEVEL OF PROMPT
4) FILE MANIPULATION
5) DIRECTORY MANIPULATION
6) OTHER SYSTEM FUNCTIONS
7) COMPILE AND EXECUTE PROGRAMS
8) TEMPORARILY EXIT MENU SHELL
9) EXIT

Enter your choice of menu options: 7

COMPILE AND EXECUTE MENU

1) RETURN TO TOP LEVEL

2) PASCAL PROGRAM

3) C PROGRAM

4) FORTRAN PROGRAM

5) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 3

C PROGRAM

You should expect a delay when your program is compiled and executed. Program compilation and execution requires more time than most of the tasks found in the menu shell.

*What is the name of the C source file you wish to compile? ?

You must enter the name of a file which contains a C program. The file name must end in ".c"

*What is the name of the C source file you wish to compile? test.c

This option will compile your C file.
*Do you also want your program executed (y or n)? ?

You will not want to have your program executed until it is free of errors. You should answer "n" to this question until you have an error free compilation.

This option will compile your C file.
*Do you also want your program executed (y or n)? y
When a C program is compiled the following options are available:

c suppress the loading phase
(Do not select this option if you wish to have your program executed.)

f floating point simulation
(Do not select this option if you are working on a VAX.)

O Optimize object code

w suppress warning diagnostics

To see the additional options use the MANUAL menu option found in the OTHER SYSTEM FUNCTIONS menu. You should ask for the "cc" command.

Enter the options you want one at a time followed by a return. When you have finished entering options simply press return.

Option (c f O w): O

Option (c f O w):

*Do you have an input file (y or n)? ?

A program which expects data to be input from the terminal can have

the data redirected from an input file. If your program is expecting data from standard input and the actual data is in a file answer y to this question.

*Do you want the output placed in a file (y or n)? ?

Any program results which are output to the terminal may be redirected to an output file. If you name an existing file as the output file the contents of that file will be replaced by the new output of your program.

*Do you want the output placed in a file (y or n)? y

What is the name of the output file? testsee

!!! cc -O test.c
!!! a.out > testsee

Press return to continue

COMPILE AND EXFCUTE MENU

1) RETURN TO TOP LEVEL

2) PASCAL PROGRAM

3) C PROGRAM

4) FORTRAN PROGRAM

5) TEMPORAPILY EXIT MENU SHELL

Enter your choice of the menu options: 5

TEMPORARILY EXIT MENU SHELL

You are able to return to the menu shell any time by pressing the return key. Proceed to enter the UNIX commands you wish to execute.

```
$ cat testsee
Output from program which demonstrates "COMPILE AND EXECUTE"
     option in the menu shell.

a=3 b=4
The product c = 12

$
```

COMPILE AND EXECUTE MENU

1) RETURN TO TOP LEVEL

2) PASCAL PROGRAM

3) C PROGRAM

4) FORTRAN PROGRAM

5) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 1

TOP IFVEI MFNU

1) INTRODUCTION TO MENU SHELL
2) INTRODUCTION TO TERMINAL
3) CHANGE LEVEL OF PROMPT
4) FILE MANIPULATION
5) DIRFCTORY MANIPULATION
6) OTHFR SYSTEM FUNCTIONS
7) COMPILE AND EXECUTE PROGRAMS
8) TEMPORARILY FXIT MENU SHFLL
9) FXIT

Enter your choice of the menu options: 3

CHANGE LEVEL OF PROMPT

*What level of prompt do you wish (1 or 2)?

There are two levels of prompting. "1" is a verbose prompt which prompts you with questions which help explain the desired task. "2" is a terse prompt which can be used once you are familiar with the available tasks.

*What level of prompt do you wish (1 or 2)? 2

TOP LEVEL MENU

1) INTRODUCTION TO MENU SHELL
2) INTRODUCTION TO TERMINAL
3) CHANGE LEVEL OF PROMPT
4) FILE MANIPULATION
5) DIRECTORY MANIPULATION
6) OTHER SYSTEM FUNCTIONS
7) COMPILE AND EXECUTE PROGRAMS
8) TEMPORARILY EXIT MENU SHELL
9) EXIT

Enter your choice of the menu option: 5

DIRECTORY MANIPULATION MENU

1) RETURN TO TOP LEVEL

2) CREATE A DIRECTORY

3) DELETE A DIRECTORY

4) LIST CONTENTS OF DIRECTORY

5) CHANGE DIRECTORY

6) PATH IDENTIFICATION

7) TEMPORARILY EXIT MENU SHELL

Enter your choice of menu options: 2

CREATE A DIRECTORY

```
*Directory name?  ?

     You must enter the name of the directory you wish to make.  A directory
name is limited to 14 characters.  Most characters may be used in a
directory name, however, until you are familiar with the special characters
and their meanings you should use only letters, numbers and the period.

*Directory name?  newdirect


!!! mkdir newdirect


Press return to continue
```

DIRECTORY MANIPULATION MENU

1) RETURN TO TOP LEVEL

2) CREATE A DIRECTORY

3) DELETE A DIRECTORY

4) LIST CONTENTS OF A DIRECTORY

5) CHANGE DIRECTORY

6) PATH IDENTIFICATION

7) TEMPORARILY EXIT MENU SHELL

Enter your choice of the menu options: 1

TOP LEVEL MENU

1) INTRODUCTION TO MENU SHELL
2) INTRODUCTION TO TERMINAL
3) CHANGE LEVEL OF PROMPT
4) FILE MANIPULATION
5) DIRECTORY MANIPULATION
6) OTHER SYSTEM FUNCTIONS
7) COMPILE AND EXECUTE PROGRAMS
8) TEMPORARILY EXIT MENU SHELL
9) EXIT

Enter your choice of the menu options: 9

BIBLIOGRAPHY

[BALL80] Ball, Eugene. Representation of Task-Specific Knowledge in a Gracefully Interacting User Interface, Report of research conducted at Carnegie-Mellon University, April, 1980.

[BARN81] Barnard, P. J. "Consistency and Compatibility in Human-Computer Dialogue," International Journal of Man-Machine Interaction, 15:87-134, 1981

[BLAC81] Black, John, and Sebrects, Marc M. "Why Interactive Computer Systems Are Sometimes Not Used by People Who Might Benefit From Them," Applied Psycholinguistics, 2:149-177, 1981.

[BLES81] Bleser, Teresa, and Foley, James D. Towards Specifying and Evaluating the Human Factors of User-computer Interfaces, Institute for Information Science and Technology, Report GWU-IIST-81-22, The George Washington University, Washington, D.C., March, 1981.

[BOTT79] Bott, R. A. A Study of Computer Learning: Theory and Methodolcgies, Center for Human Information Processing Technical Report No. 82. University of California, San Diego, 1979.

[BROW83] Brown, P. J. "Error Messages: The Neglected Area of the Man/Machine Interface? Communications of the ACM, 26:246-249, April, 1983.

[CARE82] Carey, Tom. "User Differences in Interface Design," Computer, 15:14-20, November, 1982.

[DWYE81a] Dwyer, Barry. "Programming for Users: A Bit of Psychology," Computers and People, 1:11-14,26, January, 1981.

[DWYE81b] Dwyer, Barry. "A User-Friendly Algorithm," Communications of the ACM, 24:556-561, September, 1981.

[FOLE81] Tools for the Designers of User Interfaces, Institute for Information Science and Technology, Report GWU-IIST-81-07, The George Washington University, Washington, D.C., March, 1981.

[FRIE82] Fried, Louis. "Nine Principles for Ergonomic Software," Datamation, 28:163-166, November, 1982.

[GAIN78] Gaines, Brian R. "Programming Interactive Dialogue," Pragmatic Programming & Sensible Software, 305-320. Uxbridge, England: Online Conferences Limited, 1978.

[GALI81] Galitz, Wilbert O. Handbook of Screen Format Design. Wellesley, Massachusetts: C.E.D. Information Sciences, Inc., 1981.

[GOLD80] Golden, Donald. "A Plea for Friendly Software," ACM SIGSOFT, Software Engineering Notes, 5:4-5, October, 1980.

[GRFE81] Greenburg, Robert B. "The UNIX Operating System and the Xenix Standard Operating Environment," Byte, 6:248-264, June, 1981.

[HALL78] Hall, P. Man-Computer Dialogues for Many Levels of Competence, Conference report of Information Systems Methodology, October, 1978.

[HEBD78] Hebditch, David. "The Programming Implications of Good Dialogues," Pragmatic Programming & Sensible Software", 537-547. Uxbridge, England: Online Conferences Limited, 1978.

[HEIN75] Heines, Jessie M. Style and Communication in Interactive Programming, A paper presented at the 1975 Summer Conference of the Association for the Development of Computer-based Instructional Systems, August, 1975.

[MART73] Martin, James. Design of Man-Computer Dialogues. Englewood Cliffs: Prentice Hall, Inc., 1973.

[MAYE79] Mayer, R. E. "Can Advance Organizers Influence Meaningful Learning?" Review of Educational Research, 49:372-383, 1979.

[MORA81] Moran, Thomas P. "The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems," International Journal of Man-Machine Studies, 15:3-50,1981.

[NAGE78] Nagel, Rita, and Schnupp, Peter. "Expected and Unexpected Acceptance Problems with an Interactive Programming Tool, Pragmatic Programming & Sensible Software, 321-330. Uxbridge, England: Online Conferences Limited, 1978.

[NICK81] Nickerson, Raymond. "Why Interactive Computer Systems Are Sometimes Not Used by People Who might Benefit from Them," Interactive Journal of

Man-Machine Studies, 15:469-483, 1981.

[NORM83] Norman, Donald A. "Design Rules Based on Analysis of Human Error," Communications of the ACM, 26:254-258, April, 1983.

[NORM81] Norman, Donald A. "The Trouble with UNIX," Datamation, 27:139-150, November, 1981.

[RADH82] Radhakrishnan, T., Grossner C., and Benoliel, M. "Design of an Interactive Data Retrieval System for Casual Users," Information Processing & Management,18:23-32,1982.

[RAMS79] Ramsay, Rudy H., and Atwood, Michael E. Human Factors in Computer Systems: A Review of the Literature, Englewood: Science Applications, Inc., 1979.

[RITC78] Ritchie, D. M. "UNIX Time-Sharing System: A Retrospective," The Bell System Technical Journal, 57:1947-1968, July, 1978.

[ROBE81] Robertson, G., McCracken, D., and Newell, A. "The ZOG Approach to Man-Machine Communication," International Journal of Man-Machine Studies, 14:461-488, 1981.

[SELA82] Selander, Sandy E. "Several Approaches for Improving User Cordiality in the Design of On-Line Systems for Novice Users," Computer Personnel, 9:18-21, November, 1982.

[SHNE82] Shneiderman, Ben. "Designing Computer System Messages," Communications of the ACM, 25:610-

611, September, 1982.

[SHNE80] Shneiderman, Ben. Software Psychology: Human Factors in Computer and Information Systems. Cambridge, Massachusetts: Winthrop Publishers, Inc., 1980.

[SIMP82] Simpson, Henry. "A Human-Factors Style Guide for Program Design," Byte, 7:108-132, April, 1982.

[SMIT82] Smith, David Canfield. "Designing the Star User Interface," Byte, 7:242-281, April, 1982.

[SWAI82] Swaine, Michael. "16 - Bit Revolution, Part 2: UNIX Operating System(s)," InfoWorld, 4:6-9, February 15, 1982.

[TEKA81] Tektronics. TNIX 8560 Multi-User Software Development Unit, 1981.

[WALK80] Walker, A. J. "Menu Display System," IBM Technical Disclosure Bulletin, 23:806-811, July, 1980.

[WERN82] Werner, Frank. "'User-Friendly' and Friendly Usage," Computerworld, 16:25, February, 1, 1982.

APPENDIX A

PROGRAM LISTINGS

```
/*****************************************************************/
/*  Menu shell - main program                                    */
/*                                                               */
/*      This program is a driver program to the Menu shell.      */
/*  The Menu shell is a menu driven interface to the UNIX        */
/*  operating system.                                            */
/*                                                               */
/*      Program structure:                                       */
/*                                                               */
/*        1.  Initialization process                             */
/*            a.  User introduction                              */
/*            b.  Open root menu and place on stack maintaining  */
/*                  path                                         */
/*                                                               */
/*        2.  Option processing                                  */
/*            Repeat until exit option is selected               */
/*            a.  Display menu                                   */
/*            b.  Accept user response                           */
/*            c.  Maintain path stack                            */
/*            d.  If leaf node - leaf node processing            */
/*            e.  Retrieve most recent menu                      */
/*                                                               */
/*        3.  Leaf node processing                               */
/*            a.  Execute program associated with menu option    */
/*            b.  Evaluate exit status                           */
/*                                                               */
/*      All tasks are performed in separate shell programs with  */
/*        the exception of changing directories.  A function is  */
/*        included within the main program which changes direc-  */
/*        tories.  It was necessary to include this in the main  */
/*        to assure that the directory change would be           */
/*        permanent.  A separate shell program would return to   */
/*        to the environment of the parent program and the       */
/*        directory change would be nullified.                   */
/*                                                               */
/*                                                               */
/*      The main program calls the following functions:          */
/*                                                               */
/*              intro:  presents user introduction               */
/*              choose_menu_op:  determines user response        */
/*                      to menu display                          */
/*              changedir:  performs the change directory        */
/*                      menu option                              */
/*              clear:  clears the screen                        */
/*                                                               */
/*                                                               */
/*****************************************************************/

/*
```

```
#include <stdio.h>
#include <signal.h>

#define ARRAYLEN 14
#define PATHLEN 50
#define COMMANDLEN 65
#define NUMOPT 10
#define MENUDEPTH 15
#define ROOT "m.level-1"
#define rootpath "/acct/s15/mkh1141/menu/"
#define QUESTION "?"


main()
{
        char name[ARRAYLEN];            /* file name array */
        char full_name[PATHLEN];        /* full file name array - including path from root */
        char level[ARRAYLEN];           /* file name associated with a menu option */
        char prog[ARRAYLEN];            /* name of executable program associated with menu option */
        char command[COMMANDLEN];       /* command array to forked as a child process */
        char *fileptr[NUMOPT];          /* pointers to the files associated with a menu display */
        char *stackptr[MENUDEPTH];      /* pointers maintaining path within menu structure */
        char *root_name;                /* root menu */
        char *p;
        char *malloc();

        FILE *fopen();
        FILE *fpa[MENUDEPTH];           /* array of file descriptors to the files opened in menu path */

        int c;                          /* a character read and displayed */
        int i=1;                        /* fileptr subscript */
        int j=1;                        /* stackptr subscript */
        int nomenu;                     /* number of options within a given menu */
        int choice;                     /* menu option selected */
        int leaf=0;                     /* leaf node flag:  0=non-leaf, 1=leaf */
        int cont=0;                     /* exit menu shell flag:  0=do not exit, 1=exit */
        int prompt=1;                   /* prompt level */
        int status;                     /* exit status when child process terminates */


/*
```

```
/* Ignore the break and delete key, thus preventing      */
/*   user from a non-standard exit from menu shell.      */

signal(SIGINT,SIG_IGN);



/* Initialization process                                */
/*     User introduction                                 */
                                                         */
clear();
intro();

/*      Open root menu and place on stack maintaining    */
/*        menu path                                      */
                                                         */
root_name = ROOT;
fpa[j] = fopen(root_name,"r");
p = malloc(ARRAYLEN);
strcpy(p,root_name);
stackptr[j] = p;

/*
```

```
/* Option processing                                    */

while (cont == 0)
{
/*      Display menu                                    */

        clear();
        while (leaf == 0)
        {
                c = getc(fpa[j]);                       /* get first character of file, "!" = leaf node */
                if (c != '!')
                {
                        while (c != EOF)                /* display menu if non-leaf node */
                        {
                                if (c == '#')           /* read number of menu options */
                                        fscanf(fpa[j],"%d",&nomenu);
                                else if (c == '+')      /* read file name associated with menu option */
                                {
                                        fscanf(fpa[j],"%s",level);
                                        p = malloc(ARRAYLEN);
                                        strcpy(p,level);
                                        fileptr[i++] = p;       /* record file associated with menu option */
                                }
                                else                    /* display menu one character at a time */
                                        putchar(c);
                                c = getc(fpa[j]);
                        }

                /* Accept user response                 */

                choice = choose_menu_op(nomenu);
                sprintf(full_name,"%s%s",rootpath,fileptr[choice]);
                fpa[++j] = fopen(full_name,"r");         /* open file associated with menu choice */
                p = malloc(ARRAYLEN);
                strcpy(p,fileptr[choice]);
                stackptr[j] = p;                        /* update path stack */
                i = 1;                                  /* reinitialize fileptr subscript for next menu */
                clear();
                }
        else
                leaf = 1;
        }
/*
```

```
        /* leaf node processing                                 */

        fscanf(fpa[j], "%s", prog);                             /* read executable file name */
        sprintf(command,"%s%s %d",rootpath,prog,prompt);

        /* Execute leaf file                                    */

        if (fork() == 0)
        {
                signal (SIGINT,SIG_DFL);
                execl("/bin/sh","sh","-c",command,NULL);
        }
        wait (&status);

        /* Evaluate exit status of child process                */

        switch (status)
        {
                case 256:                                       /* call change directory function */
                        changedir (prompt);
                        break;
                case 512:                                       /* return to previous menu */
                        fclose(fpa[j--]);
                        break;
                case 768:                                       /* no action taken */
                        /* If the break or delete key is pressed within an executable leaf node program, */
                        /*   the program will exit with a status of 3 (768).  The driving program takes  */
                        /*   no special action in this case.  The user is presented with the most recent */
                        /*   menu display.                                                               */
                        break;
                case 1024:                                      /* verbose prompt */
                        prompt = 1;
                        break;
                case 1280:                                      /* terse prompt */
                        prompt = 2;
                        break;
                case 1536:                                      /* exit menu shell */
                        cont = 1;
                        break;
                default:
                        break;
        }

        leaf = 0;

        /* Retrieve previous menu file                          */

        fclose(fpa[j--]);
        fclose(fpa[j]);
        sprintf(full_name,"%s%s",rootpath,stackptr[j]);
        fpa[j] = fopen(full_name, "r");


    }
}
/*
```

```
intro()

/*****************************************************************/
/*                                                               */
/* Function intro()                                              */
/*                                                               */
/*      Purpose:  To display initial welcome message to user.    */
/*                                                               */
/*              Timing is controlled with the sleep command.     */
/*              This should be modified to accommodate the       */
/*              system load.                                     */
/*                                                               */
/*****************************************************************/

{
        printf("\n\n\n\n                                    WELCOME TO THE MENU SHELL\n\n\n");

        printf("          A menu of available options will be displayed at your terminal.\n\n");

        printf("     You must type your choice of the options and the press the return\n\n");
        printf("     key.\n");
        sleep (10);

}
/*
```

```
choose_menu_op(nomenu)

/*****************************************************************/
/*                                                               */
/* Function choose_menu_op()                                     */
/*                                                               */
/*      Purpose:  To prompt user for his choice of the           */
/*                available menu options.                        */
/*                                                               */
/*      Input parameters:                                        */
/*              nomenu - number of available options in          */
/*                       current menu                            */
/*                                                               */
/*      Output parameters:                                       */
/*              choice - the number of the menu option the       */
/*                       user has selected                       */
/*                                                               */
/*      Error checking:                                          */
/*              This function continues to prompt the user       */
/*              for his menu response until he has responded     */
/*              with a numeric value within the range            */
/*              acceptable for the current menu.                 */
/*                                                               */
/*****************************************************************/

int nomenu;

{
        char garbage[50];
        int choice=0;

        while (choice <=0 || choice > nomenu)
        {
                printf("Enter your choice of the menu options:  ");
                if (scanf("%d",&choice) == 0)
                {
                        scanf("%s",garbage);
                        printf("\nOnly digits are valid menu options!  \n");
                }
                else if (choice <= 0 || choice > nomenu)
                        printf("\n%d is not a valid option!\n",choice);
        }
        return choice;
}

/*
```

```
changedir (prompt)

/*****************************************************************/
/*                                                               */
/* Function changedir(prompt)                                    */
/*                                                               */
/*      Purpose:  To provide a means of changing directories.    */
/*                                                               */
/*      Input parameters:                                        */
/*              prompt - the level of prompt                     */
/*                                                               */
/*      Error ehecking:                                          */
/*              This functions forks a shell program names       */
/*              xcheckdir which checks to see if the named       */
/*              directory ooes exist.                            */
/*                                                               */
/*      This function follows the dialogue found in the shell    */
/*      script programs.                                         */
/*                                                               */
/*****************************************************************/

int prompt;

{
        char *direct_name;              /* new directory name */
        char command[COMMANLLEN];       /* command to check if directory exists */
        char s[];                       /* accepts response to continue */

        int dstatus;                    /* status returned after directory is validated */

        direct_name = malloc(ARRAYLEN);                                 /* Initially set directory name to ? */
        strcpy(direct_name,QUESTION);
/*
```

```
        /* Present change directory dialogue */

        printf("\n\n                        CHANGE DIRECTORY\n\n");

        while (strcmp(direct_name, QUESTION) == 0)
        {
                switch (prompt)
                {
                        case 1:
                                printf("\n\n*What is the name of the directory to which you wish to change?  ");
                                break;
                        case 2:
                                printf("\n\n*Directory name ?  ");
                                break;
                }

                scanf("%s",direct_name);

                if (strcmp(direct_name,QUESTION) == 0)
                {
                        printf("\n     You must enter the full path name of the directory you wish to change\n");
                        printf("to with the following exceptions:\n");
                        printf("  1)  A child of the current directory requires only the directory name.\n");
                        printf("  2)  The parent of the current directory can be reached by entering \"..\"\n");
                        printf("  3)  A sibling of the current directory can be reached by entering\n");
                        printf("        \"../directory name\"\n");
                }
        }

        /*  Check to see if the directory name exists           */

        sprintf(command,"%s/check_dir %s",rootpath,direct_name);
        if (fork() == 0)
                execl ("/bin/sh","sh ","-c ",command,NULL);
        wait (&dstatus);

        switch (dstatus)
        {
                case 256:                                               /* directory does exist */
                        printf("\n\n!!!cd %s\n\n",direct_name);
                        chdir(direct_name);
                        printf("\n\nPress return to continue  ");
                        gets(s);
                        gets(s);
                        break;

                case 512:                                               /* directory does not exist */

                        printf("\n\nDirectory does not exist!\n\n");
                        sleep (2);
                        break;
        }
}

/*
```

```
clear()

/*************************************************************/
/*                                                           */
/* Function clear()                                          */
/*                                                           */
/*      Purpose:  To clear the screen.                       */
/*                                                           */
/*************************************************************/

{
        system("clear");
}
```

```
/*****************************************************************************/
/*  Medit.c                                                                  */
/*                                                                           */
/*  Medit.c simulates the menu structure found in the Menu shell.  It is a   */
/*    simple line oriented editor.  It provides the following options:       */
/*          1.  Display contents of a file                                   */
/*          2.  Display a screen of text beginning with a given line         */
/*          3.  Display a given line                                         */
/*          4.  Append after a given line                                    */
/*          5.  Insert before a given line                                   */
/*          6.  Delete a given line                                          */
/*          7.  Exit the editor                                              */
/*                                                                           */
/*    Medit.c spawns the UNIX editor.  The user is prompted for his choice   */
/*    of menu options.  Medit also prompts for the appropriate arguments     */
/*    for each menu option.  Medit sends the appropriate commands and        */
/*    arguments to the editor via the pipe that was opened when the editor   */
/*    process was spawned.                                                   */
/*                                                                           */
/*    The main program calls the following functions:                        */
/*        spawned:  returns the filepointer of the file used for             */
/*                  communication between medit.c and the editor process     */
/*        get_number:  returns the number of the edit option selected        */
/*                  by the user.  It includes error checking to insure       */
/*                  that the user enters a valid menu choice.                */
/*        dis_all_lines:  displays the contents of a file                    */
/*        dis_screen:  dislays a screen of text beginning with a given       */
/*                  line number.                                             */
/*        singleln:  displays a given line of a file                         */
/*        append:  append after a given line                                 */
/*        insert:  insert above a given line                                 */
/*        delete_line:  deletes a given line                                 */
/*        exit_ed:  exits the editor                                         */
/*        cont:  used to accept users response to continue edit menu         */
/*        clear:  clears the screen                                          */
/*                                                                           */
/*****************************************************************************/

/*
```

```
#include <stdio.h>
#include <signal.h>

#define ARRAYLEN 65             /* maximum length of file name to be edited */
#define MAXLINE 255             /* maximum length of line to be inserted or appended */
#define QUESTION "?"
#define DOT "."

main()
{
        char *edit_file;        /* file to be edited */

        FILE *spawned();        /* function to spawn editor */
        FILE *fptr;             /* pipe between medit.c and UNIX editor */

        int j=0;                /* flag to continue in editor:  0=edit, 1=quit editor */
        int choice;             /* number of edit option selected */
        int pid;                /* process id */


        /*  Ignore break and delete keys while in the editor.  This is an  */
        /*    exception from the other leaf node programs.  Ignoring the   */
        /*    interrupts will protect the user from exiting from the       */
        /*    without writing out the changes he has made.                 */

        signal (SIGINT,SIG_IGN);

/*
```

```
/*  Prompt user for the name of the file to be edited                */

edit_file = QUESTION;
printf ("\n\n                          EDIT A FILE\n\n");
printf ("Do not be alarmed if there is a delay in response.  The edit options\n");
printf ("require more time than most of the tasks found in the Menu shell.\n\n\n");
while (strcmp(edit_file,QUESTION) == 0)
{
        printf("\n\n*What is the name of the file you wish to edit?  ");
        scanf("%s",edit_file);
        if (strcmp(edit_file,QUESTION) == 0)
        {
                printf("\n     You must enter the name of an existing file.  If you give a new \n");
                printf("file name you will see UNIX respond with a \"? new name\".  Here you have \n");
                printf("two options:  \n");
                printf("  1)  Quit the editing session by choosing menu option 7.\n");
                printf("  2)  Append to the file by choosing menu option 4.  This will\n");
                printf("            create a new file with the new name you have given.\n\n");

                }
}


/*  Spawn the editor                                                 */


printf("\n\n!! edit %s\n\n",edit_file);
        if ((fptr = spawned(edit_file,&pid)) == NULL)
        {
                puts("Cannot spawn the editor.  Try the edit option again.\n");
                exit (0);
        }
sleep (2);

/*  Present user with edit options until user wishes to exit the     */
/*    editor                                                         */

/*
```

```
        while(j == 0)
        {
                clear();
                fflush(stdout);

                printf("\n\n                        EDIT MENU\n\n");

                printf("      1)  DISPIAY CONTENTS OF FILE\n\n");
                printf("      2)  DISPLAY A SCREEN OF TEXT BEGINNING WITH A GIVEN LINE\n\n");
                printf("      3)  DISPLAY A GIVEN LINE\n\n");
                printf("      4)  APPEND AFTER A GIVEN IINE\n\n");
                printf("      5)  INSERT BEFORF A GIVEN LINE\n\n");
                printf("      6)  DELETE A GIVEN LINE\n\n");
                printf("      7)  EXIT THE FLITOR\n\n\n\n\n\n");
                printf("Enter your choice of the menu options:  ");

                /* Accept user response to edit menu                        */

                choice = get_number();
                clear();

                /*  Call functicn associated with menu option               */

                switch (choice)
                        {
                        case 1:
                                dis_all_lines(fptr);
                                break;

                        case 2:
                                dis_screen(fptr);
                                break;

                        case 3:
                                singleln (fptr);
                                break;

                        case 4:
                                append (fptr);
                                break;

                        case 5:
                                insert (fptr);
                                break;

                        case 6:
                                delete_line(fptr);
                                break;

                        case 7:
                                exit_ed(fptr);
                                j = 1;
                                break;
                        default:
                                break;
                        }
                fflush(fptr);
                sleep (2);
        }
        exit (0);
}
```

```
FILE *spawned(file,pid)

/*****************************************************************************/
/*                                                                           */
/* Function FILE *spawned(file,pid)                                          */
/*                                                                           */
/*      Purpose:  To spawn the edit process.                                 */
/*                                                                           */
/*      Input parameters:                                                    */
/*              file - file to be edited                                     */
/*              pid  - process id                                            */
/*                                                                           */
/*      Output parameter:                                                    */
/*              fdes[1] - file descriptor for medit.c write end of pipe      */
/*                                                                           */
/*****************************************************************************/


char *file;             /* file to be edited */
int *pid;               /* process id */

{
        int tmp0;       /* temporary file descriptor */
        int fdes[2];    /* file descriptor array */

        tmp0 = dup(0);                                  /* copy standard input descriptor */
        close(0);                                       /* make descriptor 0 available */
        pipe(fdes);                                     /* open two descriptors: fdes[0] - reads */
                                                        /* from pipe, fdes[1] - write to pipe */

        switch(*pid = fork())                           /* try to spawn edit process */
        {
        case -1:
                close(fdes[0]);                         /* cannot fork process */
                close(fdes[1]);                         /* close the pipe */
                dup(tmp0);                              /* restore standard input */
                close(tmp0);                            /* close the copy */
                return(NULL);
        case 0:
                close(fdes[1]);                         /* child process - editor */
                execl("/bin/ed","ed",file,0);           /* close write end of pipe */
                exit(1);                                /* exec editor */
        default:
                close(fdes[0]);                         /* parent process - medit.c */
                dup(tmp0);                              /* close read end of pipe */
                close(tmp0);                            /* restore standard input */
                return(fdopen(fdes[1],"w"));            /* close copy of standard input */
        }                                               /* return file descriptor of the pipe */
}
/*
```

```
cont ()

/*****************************************************************************/
/*                                                                           */
/* Function cont ()                                                          */
/*                                                                           */
/*      Purpose:  To determine whether user is ready to continue exit        */
/*                process.                                                   */
/*                                                                           */
/*****************************************************************************/


{

        char s[];

        printf("\n\nPress return to continue  ");
        gets(s);
}
/*
```

```
dis_all_lines(fptr)

/*****************************************************************************/
/*                                                                           */
/* Function dis_all_lines(fptr)                                              */
/*                                                                           */
/*      Purpose:  To issue the command to the editor to display the          */
/*                contents of a file.                                        */
/*                                                                           */
/*      Input parameter:                                                     */
/*              fptr - file descriptor of pipe to editor                     */
/*                                                                           */
/*      Timing is controlled with the sleep command.  This should be         */
/*      modified to accommodate system load.  The synchronization is         */
/*      difficult to control.  Best results are obtained if very small       */
/*      files are displayed.                                                 */
/*                                                                           */
/*****************************************************************************/


FILE *fptr;

{

        printf("\n\n                                    DISPLAY CONTENTS OF FILE\n\n");
        printf("      It is possible to stop and restart the display of your file.  Remember\n");
        printf("you can temporarily stop the screen enabling you to read the display.  You\n");
        printf("can learn how to stop the display by choosing \"INTRODUCTION TO TERMINAL\"\n");
        printf("menu option at the top level menu.\n\n");
        printf("!!! 1,$p\n");
        fprintf(fptr,"1,$p\n");
        sleep (20);
}
/*
```

```
dis_screen(fptr)

/*****************************************************************************/
/*                                                                           */
/* Function dis_screen(fptr)                                                 */
/*                                                                           */
/*      Purpose:  To display one screen of text beginning with a given       */
/*                line number.  The function prompts the user for the        */
/*                line number with which he wishes to begin the display.     */
/*                                                                           */
/*      Input parameter:                                                     */
/*              fptr - file descriptor of pipe to editor                     */
/*                                                                           */
/*      Timing is controlled with the sleep command.  This should be         */
/*      adjusted to accommodate system load.                                 */
/*                                                                           */
/*****************************************************************************/


FILE *fptr;

{
        int lineno;

        char s[];

        printf("\n\n                              DISPLAY A SCREEN OF TEXT\n\n");
        printf("What line number do you wish to begin the display with?   ");
        lineno = get_number();
        printf("\n\nlll %dz\n\n",lineno);
        fprintf(fptr, "%dz\n",lineno);
        fflush(fptr);
        sleep (20);
        gets(s);
        cont();
}
/*
```

```
singleln(fptr)

/*****************************************************************************/
/*                                                                           */
/* Function singleln(fptr)                                                   */
/*                                                                           */
/*      Purpose:  To display a single line of text.  The user is prompted    */
/*                for the line number he wishes to have displayed.           */
/*                                                                           */
/*      Input parameter:                                                     */
/*              fptr - file descriptor of pipe to editor                     */
/*                                                                           */
/*      Functions called:                                                    */
/*              get_number:  returns an integer value                        */
/*              cont:  determines when user is ready to continue the         */
/*                     process                                               */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/


FILE *fptr;
{
        int lineno;

        char s[];

        printf("\n\n                              DISPLAY A GIVEN LINE\n\n");
        printf("What line number do you wish to display?  ");
        lineno = get_number();
        printf("\n\n!!! %dp\n\n",lineno);
        fprintf(fptr,"%dp\n",lineno);
        fflush(fptr);
        sleep (2);
        gets (s);
        cont ();

}
/*
```

```
append(fptr)

/*************************************************************************/
/*                                                                       */
/* Function append(fptr)                                                 */
/*                                                                       */
/*      Purpose:  To append text after a given line.  The user is        */
/*                prompted for the line number to which he wishes to     */
/*                append text.  He is also given instructions on how     */
/*                to end the append option.                              */
/*                                                                       */
/*      Input parameter:                                                 */
/*              fptr - file descriptor of pipe to editor                 */
/*                                                                       */
/*      Functions called:                                                */
/*              get_number:  returns an integer value                    */
/*              cont:  determines when user is read to continue the      */
/*                     edit process                                      */
/*                                                                       */
/*************************************************************************/


FILE *fptr;

{
        int lineno;

        char cmd[MAXLINE];

        printf("\n\n                        APPEND AFTER A GIVEN LINE\n\n");
        printf("What line number do you wish to append after?  ");
        lineno = get_number();
        printf("\n\nAfter you see the UNIX command enter the lines you wish to add after\n");
        printf("line %d.  When you have finished appending to your file you MUST enter\n",lineno);
        printf("a line which contains only a \".\"  The \".\" signals the editor that you\n");
        printf("have finished appending.\n\n");
        printf("!!! %da\n\n",lineno);
        gets (cmd);
        fprintf(fptr," %da\n",lineno);
        fflush(fptr);
        while (strcmp (cmd,DOT) != 0)
        {
                gets (cmd);
                fprintf(fptr,"%s\n",cmd);
                fflush(fptr);
        }
        sleep (2);
        cont ();
}
/*
```

```
*/


insert(fptr)

/*************************************************************************/
/*                                                                       */
/* Function insert(fptr)                                                 */
/*                                                                       */
/*      Purpose:  To insert text before a given line number.  The user   */
/*                is prompted for the line number above which he wishes  */
/*                to insert text.  He is also given instructions on how  */
/*                to end the insert option.                              */
/*                                                                       */
/*      Input parameter:                                                 */
/*              fptr - file descriptor of pipe to editor                 */
/*                                                                       */
/*      Functions called:                                                */
/*              get_number: returns an integer value                     */
/*              cont:  determines when user is read to continue the      */
/*                     edit process                                      */
/*                                                                       */
/*************************************************************************/


FILE *fptr;

{
        int lineno;

        char cmd[MAXLINE];

        printf("\n\n                          INSERT BEFORE A GIVEN LINE\n\n");
        printf("What line do you wish to insert before?  ");
        lineno = get_number();
        printf("\n\nAfter you see the UNIX command enter the lines you wish to insert before\n");
        printf("line %d.  When you have finished inserting in your file, you MUST enter\n",lineno);
        printf("a line which contains only a \".\"  The \".\" signals the editor that you\n");
        printf("have finished inserting.\n\n");
        printf("!!! %di\n",lineno);
        fprintf(fptr,"%di\n",lineno);
        gets(cmd);
        while (strcmp(cmd,EOT) !=0)
        {
                gets (cmd);
                fprintf(fptr,"%s\n",cmd);
                fflush(fptr);
        }
        sleep (2);
        cont ();
}
/*
```

```
delete_line(fptr)

/*****************************************************************************/
/*                                                                           */
/* Function delete_line(fptr)                                                */
/*                                                                           */
/*      Purpose:  To delete a given line of text.  The user is prompted      */
/*                for the number of the line he wishes to delete.            */
/*                                                                           */
/*      Input parameter:                                                     */
/*              fptr - file descriptor of pipe to editor                     */
/*                                                                           */
/*      Functions called:                                                    */
/*              get_number:  returns an integer value                        */
/*              cont:  determines when user is ready to continue the         */
/*                     edit process                                          */
/*                                                                           */
/*****************************************************************************/


FILE *fptr;
{
        int lineno;
        char s[];

        printf("\n\n                              DELETE A GIVEN LINE\n\n");
        printf("What is the number of the line you wish to delete?  ");
        lineno = get_number();
        printf("\n\n!!! %dd\n\n",lineno);
        fprintf(fptr,"%dd\n",lineno);
        fflush(fptr);
        sleep (2);
        gets(s);
        cont();
}
/*
```

```
exit_ed(fptr)

/*****************************************************************************/
/*                                                                           */
/* Function exit_ed(fptr)                                                    */
/*                                                                           */
/*      Purpose:  To close the edit session.                                 */
/*                                                                           */
/*      Input parameter:                                                     */
/*              fptr - file descriptor of pipe to editor                     */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/



FILE *fptr;

{
        int status;

        printf("\n\n                                EXIT THE EDITOR\n\n");
        printf("!!! w\n");
        printf("!!! q\n");
        fprintf(fptr,"w\n");
        fprintf(fptr,"q\n");
        fflush(fptr);
        wait (&status);
        sleep (2);
}
/*
```

```
get_number()

/*****************************************************************************/
/*                                                                           */
/* Function get_number()                                                     */
/*                                                                           */
/*      Purpose:  To accept an integer response from the user.               */
/*                                                                           */
/*      Error checking:                                                      */
/*              The user must enter a number.  The function continues        */
/*              to prompt the user for a digit until the user responds       */
/*              accordingly.                                                 */
/*                                                                           */
/*****************************************************************************/



{
        char garbage[];

        int number;

        number = -1;
        while (number == -1)
        {
                if (scanf("%d",&number) == 0)
                {
                        scanf("%s",garbage);
                        printf("\nPlease enter a digit:  ");
                }
        }
        return number;
}
/*
```

```
clear()

/*****************************************************************************/
/*                                                                           */
/* Function clear()                                                          */
/*                                                                           */
/*      Purpose:  To clear the screen.                                       */
/*                                                                           */
/*****************************************************************************/


{
        system("clear");
}
```

```
file        line level  source

xc             1     0
xc             2     0        #xc
xc             3     0        #
xc             4     0        #       Purpose - To provide a means to compile and execute a "C"
xc             5     0        #                 program.
xc             6     0        #
xc             7     0        #       Method -  The user must name the "C" program he wishes to
xc             8     0        #                 compile.  If the users presses return when first
xc             9     0        #                 asked the file name, the program will exit.  The
xc            10     0        #                 user is given 4 compile options, the option to
xc            11     0        #                 to execute the program after compilation and the
xc            12     0        #                 option for input and output files.
xc            13     0        #
xc            14     0        #       Error checking - If the user enters the name of a nonexistent
xc            15     0        #                 source file or a nonexistent input file, the user
xc            16     0        #                 is notified that the file does not exist.  The
xc            17     0        #                 program then exits.
xc            18     0        #
xc            19     0        question=?
xc            20     0        yes=y
xc            21     0        s=?             # compile option flag
xc            22     0        t=?             # program execution flag
xc            23     0        u=?             # input file flag
xc            24     0        v=?             # input file name
xc            25     0        x=?             # source file name
xc            26     0        y=?             # output file name
xc            27     0        z=?             # output file flag
xc            28     0
xc            29     0        echo '
xc            30     0
xc            31     0                                        C PROGRAM
xc            32     0
xc            33     0        You should expect a delay when your program is compiled and executed.
xc            34     0        Program compilation requires more time than most of the tasks found in
xc            35     0        the Menu shell.
xc            36     0
xc            37     0
xc            38     0
xc            39     0        #
xc            40     0        #  Request source file
xc            41     0        #
xc            42     0        while /bin/test "$x" = "$question"
xc            43     0        do
xc            44     0                echo ' '
xc            45     0                case $1 in
xc            46     0                        1)  echo -n "*What is the name of the C source file you wish to compile?  ";;
xc            47     0                        2)  echo -n "*Source file?  ";;
xc            48     0                esac
xc            49     0
xc            50     0                read x
```

```
file        line level  source

xc            51     0
xc            52     0                if /bin/test "$x"
xc            53     0                then
xc            54     0                        if /bin/test "$x" = "$question"
xc            55     0                        then
xc            56     0                                echo " "
xc            57     0                                echo "      You must enter the name of a file which contains a C"
xc            58     0                                echo "program.  The file name must end in \".c\""
xc            59     0                        fi
xc            60     0                else
xc            61     0                        exit 0
xc            62     0                        sleep 2
xc            63     0                fi
xc            64     0        done
xc            65     0
xc            66     0        #
xc            67     0        #  Determine whether program will be executed
xc            68     0        #
xc            69     0        if /bin/test -f "$x"
xc            70     0        then
xc            71     0                while /bin/test "$t" = "$question"
xc            72     0                do
xc            73     0                        echo " "
xc            74     0                        echo "This option will compile your C file."
xc            75     0                        case $1 in
xc            76     0                            1) echo -n "*Do you also want your program executed (y or n)?  ";;
xc            77     0                            2) echo -n "*Also executed (y or n)?  ";;
xc            78     0                        esac
xc            79     0
xc            80     0                        read t
xc            81     0                             .
xc            82     0
xc            83     0                        case $t in
xc            84     0                            y | Y) t=y;;
xc            85     0                            n | N) t=n;;
xc            86     0                            \?)  echo "";
xc            87     0                                 echo "      You will not want to have your program executed until it is
xc            88     0                                 echo "free of errors.  You should answer \"n\" to this question until y
xc            89     0                                 echo "have an error free compilation.";;
xc            90     0                            *)  t=?;
xc            91     0                                echo " ";;
xc            92     0                        esac
xc            93     0                done
xc            94     0        else
xc            95     0                echo
xc            96     0                echo "File $x does not exist!"
xc            97     0                sleep 2
xc            98     0                exit 0
xc            99     0        fi
xc           100     0
```

```
file            line level  source

xc               101     0
xc               102     0      #
xc               103     0      #  Request compile options
xc               104     0      #
xc               105     0      echo
xc               106     0      echo "     When a C program is compiled the following options are available:"
xc               107     0      echo "          c       suppress the loading phase"
xc               108     0      echo "                  (Do NOT select this option if you wish to have"
xc               109     0      echo "                        your program executed.)"
xc               110     0      echo "
xc               111     0      echo "          f       floating point simulation"
xc               112     0      echo "                  (Do NOT select this option if you are working on"
xc               113     0      echo "                        a VAX.)"
xc               114     0      echo "
xc               115     0      echo "          O       optimize object code"
xc               116     0      echo "
xc               117     0      echo "          w       suppress warning diagnostics"
xc               118     0      echo "
xc               119     0      echo "      To see the additional options use the MANUAL menu option found"
xc               120     0      echo "in the OTHER SYSTEM FUNCTIONS menu.  You should ask for the \"cc\""
xc               121     0      echo "command."
xc               122     0      echo
xc               123     0      echo "Enter the options you want one at a time followed by a return.  When"
xc               124     0      echo "you have finished entering options simply press return."
xc               125     0      while /bin/test "$s"
xc               126     0      do
xc               127     0              echo "
xc               128     0              echo -n "Option (c f O w):  "
xc               129     0              read s
xc               130     0              case $s in
xc               131     0                      c)      c=c;
xc               132     0                              s=?;;
xc               133     0                      f)      f=f;
xc               134     0                              s=?;;
xc               135     0                      O | o)  O=O;
xc               136     0                              s=?;;
xc               137     0                      w)      w=w;
xc               138     0                              s=?;;
xc               139     0              esac
xc               140     0      done
xc               141     0
xc               142     0      #
xc               143     0      #  Request input file
xc               144     0      #
xc               145     0      if /bin/test "$t" = "$yes
xc               146     0      then
xc               147     0              while /bin/test "$u  = "$question"
xc               148     0              do
xc               149     0                      echo "
xc               150     0                      case $1 in
```

```
file          line level  source

xc             151     0                        1) echo -n "*Do you have an input file (y or n)?  ";;
xc             152     0                        2) echo -n "*Input file (y or n)?  ";;
xc             153     0                esac
xc             154     0
xc             155     0                read u
xc             156     0
xc             157     0                case $u in
xc             158     0                        y | Y)  echo  ;
xc             159     0                                echo -n "What is the name of the input file?  ";
xc             160     0                                read v;
xc             161     0                                if  /bin/test -f  "$v"
xc             162     0                                then
xc             163     0                                        u=y
xc             164     0                                else
xc             165     0                                        echo . .
xc             166     0                                        echo "File $v does not exist!  "
xc             167     0                                        sleep 2
xc             168     0                                        exit 0
xc             169     0                                fi;;
xc             170     0                        n | N)  v= ;;
xc             171     0                        \?)     echo " ";
xc             172     0                                echo "     A program which expects data to be input from the termina
xc             173     0                                echo "the data redirected from an input file.  If your program is ex
xc             174     0                                echo "data from standard input and the actual data is in a file answ
xc             175     0                                echo "to this question.";;
xc             176     0                        *)      u=?;
xc             177     0                                echo . ";;
xc             178     0                esac
xc             179     0        done
xc             180     0
xc             181     0        #
xc             182     0        #       Request output file
xc             183     0        #
xc             184     0                while /bin/test "$z  = "$question"
xc             185     0                do
xc             186     0                        echo . .
xc             187     0                        case $1 in
xc             188     0                                1)  echo -n "*Do you want the output placed in a file (y or n)?  ";;
xc             189     0                                2)  echo -n "*Output file (y or n)?  ";;
xc             190     0                        esac
xc             191     0
xc             192     0                        read z
xc             193     0
xc             194     0                        case $z in
xc             195     0                                y | Y)  while /bin/test "$y" = "$question"
xc             196     0                                        oo
xc             197     0                                                echo .
xc             198     0                                                echo -n "*What is the name of the output file?  "
xc             199     0                                                read y
xc             200     0                                                if /bin/test $y
```

```
file            line level  source

xc               201     0
xc               202     0                                                  then
xc               203     0                                                          z=y
xc               204     0                                                  else
xc               205     0                                                          y=?
xc               206     0                                                          echo . .
xc               207     0                                                  fi
xc               208     0                                      done;;
xc               209     0                              n | N)  y= ;;
xc               210     0                              \?)     echo " ";
xc               211     0.                                     echo "      Any program results which are output to the terminal may
xc               212     0                                      echo "redirected to an output file.  If you name an existing file as
xc               213     0                                      echo "output file the contents of that file will be replaced by the
xc               214     0                                      echo "output of your program.";;
xc               215     0                              *)      z=?;
xc               216     0                                      echo . ";;
xc               217     0                      esac
xc               218     0              done
xc               219     0      fi
xc               220     0
xc               221     0      #
xc               222     0      #  Echo and execute compile command
xc               223     0      #
xc               224     0      echo . .
xc               225     0      echo -n "!!! cc "
xc               226     0      if /bin/test "$c"
xc               227     0      then
xc               228     0              echo -n "-c "
xc               229     0      fi
xc               230     0      if /bin/test "$f"
xc               231     0      then
xc               232     0              echo -n "-f "
xc               233     0      fi
xc               234     0      if /bin/test "$O"
xc               235     0      then
xc               236     0              echo -n "-O
xc               237     0      fi
xc               238     0      if /bin/test "$w"
xc               239     0      then
xc               240     0              echo -n "-w
xc               241     0      fi
xc               242     0      echo "$x"
xc               243     0      cc -$c -$f -$O -$w $x
xc               244     0      #
xc               245     0      #  Echo execution and execute program if requested
xc               246     0      #
xc               247     0
xc               248     0      if /bin/test "$t" = "$yes"
xc               249     0      then
xc               250     0              echo -n "!!!a.out "
```

```
file            line level  source

xc               251     0                  if /bin/test "$u  = "$yes'
xc               252     0                  then
xc               253     0                          echo -n '< $v '
xc               254     0                  fi
xc               255     0                  if /bin/test "$z' = "$yes'
xc               256     0                  then
xc               257     0                          echo -n '> $y'
xc               258     0                  fi
xc               259     0                  echo " "
xc               260     0                  if /bin/test "$u' = "$yes"
xc               261     0                  then
xc               262     0                          if /bin/test "$z' = "$yes'
xc               263     0                          then
xc               264     0                                  a.out < $v > $y
xc               265     0                          else
xc               266     0                                  a.out < $v
xc               267     0                          fi
xc               268     0                  else
xc               269     0                          if /bin/test "$z' = '$yes
xc               270     0                          then
xc               271     0                                  a.out > $y
xc               272     0                          else
xc               273     0                                  a.out
xc               274     0                          fi
xc               275     0                  fi
xc               276     0
xc               277     0          fi
xc               278     0
xc               279     0          echo
xc               280     0
xc               281     0          echo -n 'Press return to continue  "
xc               282     0          read y
xc               283     0
xc               284     0          exit 0
```

```
file          line level  source

xchangedir       1     0
xchangedir       2     0       #xchangedir
xchangedir       3     0       #
xchangedir       4     0       #       Purpose - To provide a means for changing directories.
xchangedir       5     0       #
xchangedir       6     0       #       Method -  This program has only one function.  It exits
xchangedir       7     0       #                 with a status of 1 which signals the main program
xchangedir       8     0       #                 to call the change directory function.
xchangedir       9     0       #
xchangedir      10     0       exit 1
```

```
file          line level  source

xchangelevel     1     0
xchangelevel     2     0       # xchangelevel
xchangelevel     3     0       #
xchangelevel     4     0       #       Purpose - To provide a means for returning to the previous
xchangelevel     5     0       #                 level in the menu structure.
xchangelevel     6     0       #
xchangelevel     7     0       #       Method -  This program has only one function.  It exits
xchangelevel     8     0       #                 with a status of 2 which signals the main program
xchangelevel     9     0       #                 to move one position within the array which
xchangelevel    10     0       #                 represents the menu structure.
xchangelevel    11     0       #
xchangelevel    12     0
xchargelevel    13     0
xchangelevel    14     0       exit 2
```

```
file           line level  source

xchangeprompt     1     0
xchangeprompt     2     0       # xchangeprompt
xchangeprompt     3     0       #
xchangeprompt     4     0       #       Purpose - To provide a means for changing the level of prompt.
xchangeprompt     5     0       #
xchangeprompt     6     0       #       Method -  When the user selects a level of prompt the program
xchangeprompt     7     0       #                 will exit with a status of 4 for the verbose prompt
xchangeprompt     8     0       #                 and 5 for the terse prompt.  If the user presses
xchangeprompt     9     0       #                 return the first time he is asked for the level
xchangeprompt    10     0       #                 of prompt no change in prompt will be made.
xchangeprompt    11     0       #
xchangeprompt    12     0       #       Error checking - The user must respond with a "1", "2" or
xchangeprompt    13     0       #                 "null".  If he responds with any other character,
xchangeprompt    14     0       #                 he will be reminded of the appropriate responses.
xchangeprompt    15     0       #
xchangeprompt    16     0       question=?
xchangeprompt    17     0       x=?
xchangeprompt    18     0       y=?
xchangeprompt    19     0
xchangeprompt    20     0       echo '
xchangeprompt    21     0
xchangeprompt    22     0                       CHANGE LEVEL OF PROMPT
xchangeprompt    23     0
xchangeprompt    24     0
xchangeprompt    25     0       while /bin/test "$x' = "$question"
xchangeprompt    26     0       do
xchangeprompt    27     0               echo " "
xchangeprompt    28     0
xchangeprompt    29     0               case $1 in
xchangeprompt    30     0                       1)  echo -n "*What level of prompt do you wish (1 or 2)?  ";;
xchangeprompt    31     0                       2)  echo -n "*Prompt level?  ";;
xchangeprompt    32     0               esac
xchangeprompt    33     0
xchangeprompt    34     0               read x
xchangeprompt    35     0
xchangeprompt    36     0               if /bin/test "$x
xchangeprompt    37     0               then
xchangeprompt    38     0                       if /bin/test "$x' = "$question'
xchangeprompt    39     0                       then
xchangeprompt    40     0                               echo " '
xchangeprompt    41     0                               echo "     There are two levels of prompting.  \"1\" is a verbose prompt whi
xchangeprompt    42     0                               echo "prompts you with questions which help explain the desired task.  \"2\"
xchangeprompt    43     0                               echo "a terse prompt which can be used once you are familiar with the availa
xchangeprompt    44     0                               echo "tasks."
xchangeprompt    45     0                       fi
xchangeprompt    46     0               else
xchangeprompt    47     0                       exit 0
xchangeprompt    48     0               fi
xchangeprompt    49     0       done
xchangeprompt    50     0
```

```
file           line level  source

xchangeprompt    51     0       while /bin/test "$y" = "$question"
xchangeprompt    52     0       do
xchangeprompt    53     0               case $x in
xchangeprompt    54     0                       1)  exit 4;;
xchangeprompt    55     0                       2)  exit 5;;
xchangeprompt    56     0                       *)  echo "";
xchangeprompt    57     0                           echo "You must enter either 1 for verbose prompts or 2 for terse prompts."
xchangeprompt    58     0                           echo -n "(1 or 2)?  ";
xchangeprompt    59     0                           read x;;
xchangeprompt    60     0               esac
xchangeprompt    61     0
xchangeprompt    62     0       done
xchangeprompt    63     0
xchangeprompt    64     0       exit 0
```

```
file          line level  source

xcheck_dir       1     0
xcheck_dir       2     0      # xcheck_dir
xcheck_dir       3     0      #
xcheck_dir       4     0      #       Purpose - This program is called by the change directory
xcheck_dir       5     0      #                 function which is called by the main program.
xcheck_dir       6     0      #                 Xcheck_dir verifies that the directory to which
xcheck_dir       7     0      #                 the user wishes to change does exist.
xcheck_dir       8     0      #
xcheck_dir       9     0      #       Method -  The program uses the "test" command to verify
xcheck_dir      10     0      #                 that the directory named in the first parameter
xcheck_dir      11     0      #                 does exist.  If the directory does exist the
xcheck_dir      12     0      #                 program exits with a status of 1 and if the
xcheck_dir      13     0      #                 directory does not exist the program exits with
xcheck_dir      14     0      #                 a status of 2.
xcheck_dir      15     0      #
xcheck_dir      16     0
xcheck_dir      17     0      if /bin/test -d $1
xcheck_dir      18     0      then
xcheck_dir      19     0              exit 1
xcheck_dir      20     0      else
xcheck_dir      21     0              exit 2
xcheck_dir      22     0      fi
```

```
file        line level  source

xcopyfile      1     0
xcopyfile      2     0    # xcopyfile
xcopyfile      3     0    #
xcopyfile      4     0    #       Purpose - To provide a means for copying a file.
xcopyfile      5     0    #
xcopyfile      6     0    #       Method -  The user must enter the name of the file he wishes
xcopyfile      7     0    #                 to have copied and the name of the file where the
xcopyfile      8     0    #                 copy will be placed.  If the user presses return
xcopyfile      9     0    #                 when first asked for the source file, the program
xcopyfile     10     0    #                 will exit.
xcopyfile     11     0    #
xcopyfile     12     0    #       Error checking - If the user gives the name of a nonexistent
xcopyfile     13     0    #                 file to be copied, he will told the file does not
xcopyfile     14     0    #                 exist and the program will exit.
xcopyfile     15     0    #
xcopyfile     16     0    question=?
xcopyfile     17     0    x=?
xcopyfile     18     0    y=?
xcopyfile     19     0
xcopyfile     20     0    echo '
xcopyfile     21     0
xcopyfile     22     0         .                          COPY A FILE
xcopyfile     23     0
xcopyfile     24     0
xcopyfile     25     0    while /bin/test "$x" = "$question"
xcopyfile     26     0    do
xcopyfile     27     0            echo " "
xcopyfile     28     0            case $1 in
xcopyfile     29     0                    1) echo -n "*What is the name of the file you wish to copy?  ";;
xcopyfile     30     0                    2) echo -n "*Source file?  ";;
xcopyfile     31     0            esac
xcopyfile     32     0
xcopyfile     33     0            read x
xcopyfile     34     0
xcopyfile     35     0            if test "$x'
xcopyfile     36     0            then
xcopyfile     37     0
xcopyfile     38     0                    if test "$x' = "$question"
xcopyfile     39     0                    then
xcopyfile     40     0                            echo " '
xcopyfile     41     0                            echo '     You must enter the name of an existing file.  If the file is not'
xcopyfile     42     0                            echo "in the current directory the entire path name must be entered."
xcopyfile     43     0                    fi
xcopyfile     44     0            else
xcopyfile     45     0                    exit 0
xcopyfile     46     0            fi
xcopyfile     47     0    done
xcopyfile     48     0
xcopyfile     49     0    if /bin/test -f $x
xcopyfile     50     0    then
```

```
file        line level  source

xcopyfile     51    0              while /bin/test "$y  = "$question"
xcopyfile     52    0              do
xcopyfile     53    0                      echo " "
xcopyfile     54    0                      case $1 in
xcopyfile     55    0                              1)  echo -n "*Where do you wish to copy the file?  ";;
xcopyfile     56    0                              2)  echo -n "*Destination file?  ";;
xcopyfile     57    0                      esac
xcopyfile     58    0
xcopyfile     59    0                      read y
xcopyfile     60    0                      if /bin/test "$y"
xcopyfile     61    0                      then
xcopyfile     62    0                              if /bin/test "$y" = "$question"
xcopyfile     63    0                              then
xcopyfile     64    0                                      echo " "
xcopyfile     65    0                                      echo "
xcopyfile     66    0                                      echo "     You must enter a new file name.  If you enter the name of
xcopyfile     67    0                                      echo "file the contents of that file will be lost.  The file will be
xcopyfile     68    0                                      echo "with the contents of $x."
xcopyfile     69    0                              fi
xcopyfile     70    0                      else
xcopyfile     71    0                              echo "
xcopyfile     72    0                              echo "     You must enter the name of the file where $x will be copied.
xcopyfile     73    0                              y=?
xcopyfile     74    0                      fi
xcopyfile     75    0              done
xcopyfile     76    0
xcopyfile     77    0              echo "
xcopyfile     78    0                    "
xcopyfile     79    0              echo "!!! cp $x $y"
xcopyfile     80    0
xcopyfile     81    0              cp $x $y
xcopyfile     82    0
xcopyfile     83    0              echo "
xcopyfile     84    0                    "
xcopyfile     85    0              echo -n "Press return to continue  "
xcopyfile     86    0              read z
xcopyfile     87    0      else
xcopyfile     88    0              echo "
xcopyfile     89    0              echo "File $x does not exist!"
xcopyfile     90    0              sleep 2
xcopyfile     91    0
xcopyfile     92    0      fi
xcopyfile     93    0
xcopyfile     94    0      exit 0
```

```
file        line level  source
xcreatedir     1     0
xcreatedir     2     0    #xcreatedir
xcreatedir     3     0    #
xcreatedir     4     0    #       Purpose - To provide a means for creating a directory.
xcreatedir     5     0    #
xcreatedir     6     0    #       Method -  The user must name the directory he wishes to
xcreatedir     7     0    #                 create.  If the user presses return when first
xcreatedir     8     0    #                 asked for the directory name, the program will
xcreatedir     9     0    #                 exit.
xcreatedir    10     0    #
xcreatedir    11     0    #       Error checking - If the user enters the name of an
xcreatedir    12     0    #                 existing directory, he will be told the
xcreatedir    13     0    #                 directory exists and the program will exit.
xcreatedir    14     0    #
xcreatedir    15     0
xcreatedir    16     0    question=?
xcreatedir    17     0    x=?
xcreatedir    18     0
xcreatedir    19     0    echo '
xcreatedir    20     0
xcreatedir    21     0                          CREATE A DIRECTORY
xcreatedir    22     0
xcreatedir    23     0                    '
xcreatedir    24     0    while /bin/test "$x" = "$question"
xcreatedir    25     0    do
xcreatedir    26     0            echo " '
xcreatedir    27     0            case $1 in
xcreatedir    28     0                    1)  echo -n "*What is the name of the directory you wish to create?  ";;
xcreatedir    29     0                    2)  echo -n "*Directory name?  ";;
xcreatedir    30     0            esac
xcreatedir    31     0
xcreatedir    32     0            read x
xcreatedir    33     0
xcreatedir    34     0            if /bin/test "$x'
xcreatedir    35     0            then
xcreatedir    36     0                    if /bin/test "$x' = "$question"
xcreatedir    37     0                    then
xcreatedir    38     0                            echo " "
xcreatedir    39     0                            echo '      You must enter the name of the directory you wish to make.  A dir
xcreatedir    40     0                            echo 'name is limited to 14 characters.  Most characters may be used in a"
xcreatedir    41     0                            echo "directory name, however, until you are familiar with the special chara
xcreatedir    42     0                            echo "and their meanings you should use only letters, numbers and the period
xcreatedir    43     0                    fi
xcreatedir    44     0            else
xcreatedir    45     0                    exit 0
xcreatedir    46     0            fi
xcreatedir    47     0    done
xcreatedir    48     0
xcreatedir    49     0    if /bin/test -d $x
xcreatedir    50     0    then
```

```
file        line level  source

xcreatedir    51    0                 echo ' '
xcreatedir    52    0                 echo "Directory $x already exists!"
xcreatedir    53    0
xcreatedir    54    0                 sleep 2
xcreatedir    55    0           else
xcreatedir    56    0                 echo '
xcreatedir    57    0                 "
xcreatedir    58    0                 echo "!!! mkdir $x'
xcreatedir    59    0
xcreatedir    60    0                 mkdir $x
xcreatedir    61    0
xcreatedir    62    0                 echo "
xcreatedir    63    0                 "
xcreatedir    64    0                 echo -n "Press return to continue  "
xcreatedir    65    0                 read z
xcreatedir    66    0           fi
xcreatedir    67    0
xcreatedir    68    0           exit 0
```

```
file          line level  source

xcreatefile      1     0
xcreatefile      2     0       # xcreatefile
xcreatefile      3     0       #
xcreatefile      4     0       #       Purpose - To provide a means for creating a new file.
xcreatefile      5     0       #
xcreatefile      6     0       #       Method -  The program uses the "cat" command to create a
xcreatefile      7     0       #                 new file.  If the user presses return in response
xcreatefile      8     0       #                 to the file name this program immediately exits.
xcreatefile      9     0       #
xcreatefile     10     0       #       Error checking - If the user gives the file a name which
xcreatefile     11     0       #                 already exists, the  program informs the user and
xcreatefile     12     0       #                 exits.
xcreatefile     13     0       #
xcreatefile     14     0       question=?
xcreatefile     15     0       x=?
xcreatefile     16     0
xcreatefile     17     0       echo '
xcreatefile     18     0
xcreatefile     19     0                                   CREATE A FILE
xcreatefile     20     0
xcreatefile     21     0            '
xcreatefile     22     0
xcreatefile     23     0       while /bin/test "$x" = "$question"
xcreatefile     24     0       do
xcreatefile     25     0               echo ' '
xcreatefile     26     0               case $1 in
xcreatefile     27     0                       1)  echo -n "*What is the name of the file you wish to create?    ";;
xcreatefile     28     0                       2)  echo -n "*File name?  ";;
xcreatefile     29     0               esac
xcreatefile     30     0
xcreatefile     31     0               read x
xcreatefile     32     0
xcreatefile     33     0               if /bin/test "$x"
xcreatefile     34     0               then
xcreatefile     35     0                       if /bin/test "$x  = "$question
xcreatefile     36     0                       then
xcreatefile     37     0                               echo " '
xcreatefile     38     0                               echo "      A file name is limited to 14 characters.  Most characters may be
xcreatefile     39     0                               echo "a file name, however until you are familiar with the special character
xcreatefile     40     0                               echo "their meanings you should use only letters, numbers and the period."
xcreatefile     41     0                       fi
xcreatefile     42     0               else
xcreatefile     43     0                       exit 0
xcreatefile     44     0               fi
xcreatefile     45     0       done
xcreatefile     46     0
xcreatefile     47     0       if /bin/test -f $x
xcreatefile     48     0       then
xcreatefile     49     0               echo
xcreatefile     50     0               echo  File $x already exists!"
```

```
file           line level  source

xcreatefile      51     0              sleep 2
xcreatefile      52     0
xcreatefile      53     0      else
xcreatefile      54     0              echo '
xcreatefile      55     0
xcreatefile      56     0              echo "After seeing the UNIX command, enter the contents of your file.  You
xcreatefile      57     0              echo "are able to make corrections on the current line you are typing, however
xcreatefile      58     0              echo "once you press the return key for a given line, corrections must be done
xcreatefile      59     0              echo "at a later time by chcosing the \"EDIT A FILE\" menu option."
xcreatefile      60     0              echo
xcreatefile      61     0              echo "TO INDICATE THAT YOU RAVE FINISHED ENTERING YOUR FILE PRESS THE \"CONTROL\""
xcreatefile      62     0              echo "KEY AND THE \"D\" KEY SIMULTANEOUSLY."
xcreatefile      63     0              echo "
xcreatefile      64     0
xcreatefile      65     0              echo "!!! cat > $x"
xcreatefile      66     0              echo
xcreatefile      67     0                   '
xcreatefile      68     0
xcreatefile      69     0
xcreatefile      70     0              cat > $x
xcreatefile      71     0
xcreatefile      72     0              echo '
xcreatefile      73     0                   '
xcreatefile      74     0              echo -n "Press return to continue  '
xcreatefile      75     0              read z
xcreatefile      76     0
xcreatefile      77     0      fi
xcreatefile      78     0
xcreatefile      79     0      exit 0
```

```
file        line level  source

xdatetime      1     0
xdatetime      2     0
xdatetime      3     0       #xdatetime
xdatetime      4     0       #
xdatetime      5     0       #       Purpose - To provide a means for printing the current date
xdatetime      6     0       #                 and time.
xdatetime      7     0       #
xdatetime      8     0       #       Method -  A brief explanation of the "date' command is
xdatetime      9     0       #                 given.  The shell then issues the "date" command.
xdatetime     10     0       #
xdatetime     11     0       echo '
xdatetime     12     0
xdatetime     13     0                               CURRENT DATE AND TIME
xdatetime     14     0                       '
xdatetime     15     0
xdatetime     16     0            The following command returns the current date and time:
xdatetime     17     0
xdatetime     18     0
xdatetime     19     0       !!! date
xdatetime     20     0           '
xdatetime     21     0
xdatetime     22     0
xdatetime     23     0       date
xdatetime     24     0
xdatetime     25     0       echo '
xdatetime     26     0       echo -n "Press return to continue  '
xdatetime     27     0       read z
xdatetime     28     0
xdatetime     29     0       exit 0
```

```
file        line level  source

xdeletedir     1     0
xdeletedir     2     0       #xdeletedir
xdeletedir     3     0       #
xdeletedir     4     0       #       Purpose - To provide a means for deleting a directory.
xdeletedir     5     0       #
xdeletedir     6     0       #       Method -  The user must name the directory he wishes to
xdeletedir     7     0       #                 to delete.  If the user presses return when
xdeletedir     8     0       #                 first asked for the directory name, the program
xdeletedir     9     0       #                 will exit.
xdeletedir    10     0       #
xdeletedir    11     0       #       Error checking - If the user enters the name of a nonexistent
xdeletedir    12     0       #                 directory, the program will tell him it does not
xdeletedir    13     0       #                 exist and then exit.
xdeletedir    14     0       #
xdeletedir    15     0       question=?
xdeletedir    16     0       x=?
xdeletedir    17     0       echo
xdeletedir    18     0
xdeletedir    19     0                               DELETE A DIRECTORY
xdeletedir    20     0
xdeletedir    21     0       .
xdeletedir    22     0       while /bin/test "$x" = "$question
xdeletedir    23     0       do
xdeletedir    24     0               echo    .
xdeletedir    25     0               case $1 in
xdeletedir    26     0                       1)  echo -n "*What is the name of the directory you wish to delete?  ";;
xdeletedir    27     0                       2)  echo -n "*Directory name?  ";;
xdeletedir    28     0               esac
xdeletedir    29     0
xdeletedir    30     0               read x
xdeletedir    31     0
xdeletedir    32     0               if /bin/test "$x"
xdeletedir    33     0               then
xdeletedir    34     0                       if /bin/test "$x  =  $question"
xdeletedir    35     0                       then
xdeletedir    36     0                               echo .
xdeletedir    37     0                               echo "
xdeletedir    38     0                               echo "     You must enter the name of an existing directory.  The directory"
xdeletedir    39     0                               echo "must te empty before it can be deleted."
xdeletedir    40     0                       fi
xdeletedir    41     0               else
xdeletedir    42     0                       exit 0
xdeletedir    43     0               fi
xdeletedir    44     0       done
xdeletedir    45     0
xdeletedir    46     0       if /bin/test -d $x
xdeletedir    47     0       then
xdeletedir    48     0               echo
xdeletedir    49     0
xdeletedir    50     0               echo "!!!rmdir $x
```

```
file          line level  source

xdeletedir      51     0                    .
xdeletedir      52     0
xdeletedir      53     0              rmdir $x
xdeletedir      54     0
xdeletedir      55     0              echo '
xdeletedir      56     0              .
xdeletedir      57     0
xdeletedir      58     0              echo -n "Press return to continue
xdeletedir      59     0
xdeletedir      60     0              read z
xdeletedir      61     0       else
xdeletedir      62     0              echo   .
xdeletedir      63     0              echo "Directory $x does not exist!"
xdeletedir      64     0              sleep 2
xdeletedir      65     0       fi
xdeletedir      66     0
xdeletedir      67     0       exit 0
```

```
file          line level  source

xdeletefile      1     0
xdeletefile      2     0      # xdeletefile
xdeletefile      3     0      #
xdeletefile      4     0      #       Purpose - To provide a means for deleting a file.
xdeletefile      5     0      #
xdeletefile      6     0      #       Method -  The user must name the file he wishes to delete.
xdeletefile      7     0      #                 If the user presses return when first asked for
xdeletefile      8     0      #                 name of the file he wishes to delete, the program
xdeletefile      9     0      #                 will exit.
xdeletefile     10     0      #
xdeletefile     11     0      #       Error checking:  If the users asks to delete a nonexistent
xdeletefile     12     0      #                 file, the program will tell him it does not exist
xdeletefile     13     0      #                 and exit.
xdeletefile     14     0      #
xdeletefile     15     0      question=?
xdeletefile     16     0      x=?
xdeletefile     17     0
xdeletefile     18     0      echo '
xdeletefile     19     0
xdeletefile     20     0                                  DELETE A FILE
xdeletefile     21     0
xdeletefile     22     0
xdeletefile     23     0      while /bin/test "$x" = "$question"
xdeletefile     24     0      do
xdeletefile     25     0              echo ' "
xdeletefile     26     0              case $1 in
xdeletefile     27     0                      1)  echo -n "*What is the name of the file you wish to delete?  ";;
xdeletefile     28     0                      2)  echo -n "*File name?  ";;
xdeletefile     29     0              esac
xdeletefile     30     0
xdeletefile     31     0              read x
xdeletefile     32     0
xdeletefile     33     0              if /bin/test "$x"
xdeletefile     34     0              then
xdeletefile     35     0                      if /bin/test "$x" = "$question"
xdeletefile     36     0                      then
xdeletefile     37     0                              echo " "
xdeletefile     38     0                              echo "     You must enter the name of an existing file.  The file you name"
xdeletefile     39     0                              echo "will te permanently removed.  Remember if you change your mind and"
xdeletefile     40     0                              echo "no longer wish to delete a file, simply press the return key."
xdeletefile     41     0                      fi
xdeletefile     42     0              else
xdeletefile     43     0                      exit 0
xdeletefile     44     0              fi
xdeletefile     45     0      done
xdeletefile     46     0
xdeletefile     47     0      if /bin/test -f $x
xdeletefile     48     0      then
xdeletefile     49     0              echo
xdeletefile     50     0
```

```
file          line level  source

xdeletefile     51     0              echo "!!! rm $x
xdeletefile     52     0
xdeletefile     53     0              rm $x
xdeletefile     54     0
xdeletefile     55     0              echo '
xdeletefile     56     0              '
xdeletefile     57     0              echo -n "Press return to continue '
xdeletefile     58     0
xdeletefile     59     0              read z
xdeletefile     60     0
xdeletefile     61     0      else
xdeletefile     62     0              echo ' '
xdeletefile     63     0              echo "File $x does not exist!"
xdeletefile     64     0              sleep 2
xdeletefile     65     0
xdeletefile     66     0      fi
xdeletefile     67     0
xdeletefile     68     0      exit 0
```

```
file          line level  source

xdisplayfile     1     0
xdisplayfile     2     0       # xdisplayfile
xdisplayfile     3     0       #
xdisplayfile     4     0       #       Purpose - To provide a means for a scrolling display of a file.
xdisplayfile     5     0       #
xdisplayfile     6     0       #       Method -  The user enters the name of the file he wishes to
xdisplayfile     7     0       #                 display.  If the user presses return when first asked
xdisplayfile     8     0       #                 for the name of the file to be displayed, the program
xdisplayfile     9     0       #                 will exit.
xdisplayfile    10     0       #
xdisplayfile    11     0       #       Error checking - If the users enters the name of a nonexistent
xdisplayfile    12     0       #                 file, the program will tell him it does not exist
xdisplayfile    13     0       #                 and exit
xdisplayfile    14     0       #
xdisplayfile    15     0       question=?
xdisplayfile    16     0       x=?
xdisplayfile    17     0
xdisplayfile    18     0       echo '
xdisplayfile    19     0
xdisplayfile    20     0                       SCROLLED DISPLAY OF A FILE
xdisplayfile    21     0
xdisplayfile    22     0            "
xdisplayfile    23     0
xdisplayfile    24     0       while /bin/test "$x" = "$question"
xdisplayfile    25     0       do
xdisplayfile    26     0               echo ' "
xdisplayfile    27     0               case $1 in
xdisplayfile    28     0                       1)  echo -n "*What is the name of the file you wish to display?  ";;
xdisplayfile    29     0                       2)  echo -n "*File name?  ";;
xdisplayfile    30     0               esac
xdisplayfile    31     0
xdisplayfile    32     0               read x
xdisplayfile    33     0
xdisplayfile    34     0               if /bin/test "$x"
xdisplayfile    35     0               then
xdisplayfile    36     0                       if /bin/test "$x" = "$question"
xdisplayfile    37     0                       then
xdisplayfile    38     0                               echo ' '
xdisplayfile    39     0                               echo '      You must enter the name of an existing file.  Remember you are'
xdisplayfile    40     0                               echo "to stop the screen enabling you to read the display.  You can learn"
xdisplayfile    41     0                               echo "how to stop the display by choosing the \"INTRODUCTION TO TERMINAL\""
xdisplayfile    42     0                               echo "menu option at the top level menu.
xdisplayfile    43     0                       fi
xdisplayfile    44     0               else
xdisplayfile    45     0                       exit 0
xdisplayfile    46     0               fi
xdisplayfile    47     0       done
xdisplayfile    48     0
xdisplayfile    49     0       if /bin/test -f "$x"
xdisplayfile    50     0       then
```

```
file           line level  source

xdisplayfile     51     0              echo "
xdisplayfile     52     0                   '
xdisplayfile     53     0              echo "!!! cat $x'
xdisplayfile     54     0              echo "
xdisplayfile     55     0                   '
xdisplayfile     56     0
xdisplayfile     57     0              cat $x
xdisplayfile     58     0
xdisplayfile     59     0              echo '
xdisplayfile     60     0                   "
xdisplayfile     61     0              echo -n "Press return to continue  "
xdisplayfile     62     0              read z
xdisplayfile     63     0       else
xdisplayfile     64     0              echo " "
xdisplayfile     65     0              echo "File $x does not exist!"
xdisplayfile     66     0              sleep 2
xdisplayfile     67     0       fi
xdisplayfile     68     0
xdisplayfile     69     0       exit 0
```

```
file          line level  source

xexit            1     0
xexit            2     0       # xexit
xexit            3     0       #
xexit            4     0       #       Purpose - To provide a means for exiting the Menu shell.
xexit            5     0       #
xexit            6     0       #       Method -  This program has only one function.  It exits
xexit            7     0       #                 with a status of 6 which signals the main program
xexit            8     0       #                 to exit the Menu shell.
xexit            9     0       #
xexit           10     0
xexit           11     0
xexit           12     0       exit 6
```

```
file        line level  source

xfortran       1     0
xfortran       2     0       #xfortran
xfortran       3     0       #
xfortran       4     0       #       Purpose - To provide a means to compile and execute a fortran
xfortran       5     0       #                 program.
xfortran       6     0       #
xfortran       7     0       #       Method -  The user must enter the name of the fortran program
xfortran       8     0       #                 he wishes to compile.  If he presses return when
xfortran       9     0       #                 first asked, the program will exit.  The user is
xfortran      10     0       #                 given the option to execute the program after
xfortran      11     0       #                 compilation.  He is also given the option for
xfortran      12     0       #                 3 compile options.  The user may use input
xfortran      13     0       #                 and output files.
xfortran      14     0       #
xfortran      15     0       #       Error checking - If the user names a nonexistent fortran
xfortran      16     0       #                 source file or a nonexistent input file, the
xfortran      17     0       #                 user will be notified that the file does not
xfortran      18     0       #                 exist and the program will exit.
xfortran      19     0       #
xfortran      20     0       question=?
xfortran      21     0       yes=y
xfortran      22     0       s=?             #  Compile options flag
xfortran      23     0       t=?             #  Program execution flag
xfortran      24     0       u=?             #  Input file flag
xfortran      25     0       v=?             #  Input file name
xfortran      26     0       x=?             #  Source file name
xfortran      27     0       y=?             #  Output file name
xfortran      28     0       z=?             #  Output file flag
xfortran      29     0
xfortran      30     0       echo '
xfortran      31     0
xfortran      32     0                               FORTRAN PROGRAM
xfortran      33     0
xfortran      34     0       You should expect a delay when your program is compiled and executed.
xfortran      35     0       Program compilation requires more time than most of the tasks found in
xfortran      36     0       the Menu shell.
xfortran      37     0
xfortran      38     0
xfortran      39     0            '
xfortran      40     0
xfortran      41     0       #
xfortran      42     0       #  Request source file
xfortran      43     0       #
xfortran      44     0       while /bin/test "$x" = "$question"
xfortran      45     0       do
xfortran      46     0               echo " "
xfortran      47     0               case $1 in
xfortran      48     0                       1)  echo -n "*What is the name of FORTRAN source file you wish to compile?   ;;
xfortran      49     0                       2)  echo -n "*Source file?  ";;
xfortran      50     0               esac
```

```
file        line level  source

xfortran      51     0
xfortran      52     0                read x
xfortran      53     0
xfortran      54     0                if /bin/test "$x"
xfortran      55     0                then
xfortran      56     0                        if /bin/test "$x" = "$question"
xfortran      57     0                        then
xfortran      58     0                                echo
xfortran      59     0                                echo "      You must enter the name of a file which contains a FORTRAN"
xfortran      60     0                                echo "program.  The file name must end in \".f\"
xfortran      61     0                        fi
xfortran      62     0                else
xfortran      63     0                        exit 0
xfortran      64     0                fi
xfortran      65     0        done
xfortran      66     0
xfortran      67     0        #
xfortran      68     0        #  Determine whether program will also be executed
xfortran      69     0        #
xfortran      70     0        if /bin/test -f "$x"
xfortran      71     0        then
xfortran      72     0                while /bin/test "$t" = "$question"
xfortran      73     0                do
xfortran      74     0                        echo ' '
xfortran      75     0                        echo "This option will compile your FORTRAN file."
xfortran      76     0                        case $1 in
xfortran      77     0                                1) echo -n "*Do you also want your program executed (y or n)?  ";;
xfortran      78     0                                2) echo -n "*Also executed (y or n)?  ";;
xfortran      79     0                        esac
xfortran      80     0
xfortran      81     0                        read t
xfortran      82     0
xfortran      83     0
xfortran      84     0                        case $t in
xfortran      85     0                                y | Y) t=y;;
xfortran      86     0                                n | N) t=n;;
xfortran      87     0                                \?)  echo " ";
xfortran      88     0                                     echo "      You will not want to have your program executed until it is
xfortran      89     0                                     echo "free of errors.  You should answer "no" to this question until yo
xfortran      90     0                                     echo "have an error free compilation.";;
xfortran      91     0                                *)  t=?;
xfortran      92     0                                    echo " ";;
xfortran      93     0                        esac
xfortran      94     0                done
xfortran      95     0        else
xfortran      96     0                echo ' '
xfortran      97     0                echo 'File $x does not exist!'
xfortran      98     0                sleep 2
xfortran      99     0                exit 0
xfortran     100     0        fi
```

```
file        line level  source

xfortran     101     0
xfortran     102     0
xfortran     103     0      #
xfortran     104     0      #  Request compile options
xfortran     105     0      #
xfortran     106     0      echo
xfortran     107     0      echo '      When a FORTRAN program is compiled the following options are available:'
xfortran     108     0      echo '           c       suppress the loading phase"
xfortran     109     0      echo                         (Do NOT select this option if you wish to have"
xfortran     110     0      echo                               your program executed.)"
xfortran     111     0      echo
xfortran     112     0      echo '           f       floating point simulation
xfortran     113     0      echo                         (Do NOT select this option if you are working on'
xfortran     114     0      echo                               a VAX.)"
xfortran     115     0      echo
xfortran     116     0      echo '           w       suppress warning diagnostics
xfortran     117     0      echo
xfortran     118     0      echo '      To see the additional options use the MANUAL menu item in the'
xfortran     119     0      echo "top level menu OTHER SYSTEM FUNCTIONS.  You should ask for the \'f77\"'
xfortran     120     0      echo "command."
xfortran     121     0      echo "
xfortran     122     0      echo "Enter the options you want one at a time followed by a return.  When'
xfortran     123     0      echo "you have finished entering options simply press return."
xfortran     124     0      while /bin/test "$s"
xfortran     125     0      do
xfortran     126     0              echo ' "
xfortran     127     0              echo -n "Option (c f w):  "
xfortran     128     0              read s
xfortran     129     0              case $s in
xfortran     130     0                      c)      c=c;
xfortran     131     0                              s=?;;
xfortran     132     0                      f)      f=f;
xfortran     133     0                              s=?;;
xfortran     134     0                      w)      w=w;
xfortran     135     0                              s=?;;
xfortran     136     0              esac
xfortran     137     0      done
xfortran     138     0
xfortran     139     0      #
xfortran     140     0      #  Request input file
xfortran     141     0      #
xfortran     142     0      if /bin/test "$t" = "$yes'
xfortran     143     0      then
xfortran     144     0              while /bin/test "$u  = "$question'
xfortran     145     0              do
xfortran     146     0                      echo ' '
xfortran     147     0                      case $1 in
xfortran     148     0                              1) echo -n "*Do you have an input file (y or n)?   ;;
xfortran     149     0                              2) echo -n "*Input file (y or n)?   ;;
xfortran     150     0                      esac
```

```
file        line level  source

xfortran     151    0
xfortran     152    0                       read u
xfortran     153    0
xfortran     154    0                       case $u in
xfortran     155    0                               y | Y)  echo ' ';
xfortran     156    0                                       echo -n "What is the name of the input file?   ";
xfortran     157    0                                       read v;
xfortran     158    0                                       if  /bin/test -f  "$v
xfortran     159    0                                       then
xfortran     160    0                                               u=y
xfortran     161    0                                       else
xfortran     162    0                                               echo ' '
xfortran     163    0                                               echo "File $v does not exist!  "
xfortran     164    0                                               sleep 2
xfortran     165    0                                               exit 0
xfortran     166    0                                       fi;;
xfortran     167    0                               n | N)  v= ;;
xfortran     168    0                               \?)     echo "  ";
xfortran     169    0                                       echo "      A program which expects data to be input from the termina
xfortran     170    0                                       echo "the data redirected from an input file.  If your program is ex
xfortran     171    0                                       echo "data from standard input and the actual data is in a file answ
xfortran     172    0                                       echo "to this question.";;
xfortran     173    0                               *)      u=?;
xfortran     174    0                                       echo   ";;
xfortran     175    0                       esac
xfortran     176    0               done
xfortran     177    0
xfortran     178    0       #
xfortran     179    0       #       Request output file
xfortran     180    0       #
xfortran     181    0               while /bin/test "$z" = "$question"
xfortran     182    0               do
xfortran     183    0                       echo ' '
xfortran     184    0                       case $1 in
xfortran     185    0                               1)  echo -n "*Do you want the output placed in a file (y or n)?   ";;
xfortran     186    0                               2)  echo -n "*Output file (y or n)?  ";;
xfortran     187    0                       esac
xfortran     188    0
xfortran     189    0                       read z
xfortran     190    0
xfortran     191    0                       case $z in
xfortran     192    0                               y | Y)  while /bin/test "$y" = '$question'
xfortran     193    0                                       do
xfortran     194    0                                               echo ' '
xfortran     195    0                                               echo -n "*What is the name of the output file?  "
xfortran     196    0                                               read y
xfortran     197    0                                               if /bin/test $y
xfortran     198    0                                               then
xfortran     199    0                                                       z=y
xfortran     200    0                                               else
```

```
file        line level  source

xfortran     201    0
xfortran     202    0                                                    y=?
xfortran     203    0                                                    echo
xfortran     204    0                                                fi
xfortran     205    0                                            done;;
xfortran     206    0                                    n | N)  y= ;;
xfortran     207    0                                    \?)     echo " ";
xfortran     208    0                                            echo "      Any program results which are output to the terminal may
xfortran     209    0                                            echo "redirected to an output file.  If you name an existing file as
xfortran     210    0                                            echo "output file the contents of that file will be replaced by the
xfortran     211    0                                            echo "output of your program.";;
xfortran     212    0                                    *)      z=?;
xfortran     213    0                                            echo " ";;
xfortran     214    0                            esac
xfortran     215    0                    done
xfortran     216    0            fi
xfortran     217    0
xfortran     218    0            #
xfortran     219    0            #  Echo and execute compile commands
xfortran     220    0            #
xfortran     221    0            echo "
xfortran     222    0            echo -n  !!! f77 "
xfortran     223    0            if /bin/test "$c"
xfortran     224    0            then
xfortran     225    0                    echo -n "-c "
xfortran     226    0            fi
xfortran     227    0            if /bin/test "$f"
xfortran     228    0            then
xfortran     229    0                    echo -n "-f "
xfortran     230    0            fi
xfortran     231    0            if /bin/test "$w"
xfortran     232    0            then
xfortran     233    0                    echo -n "-w "
xfortran     234    0            fi
xfortran     235    0            echo  "$x "
xfortran     236    0
xfortran     237    0            if /bin/test "$c"
xfortran     238    0            then
xfortran     239    0                    if /bin/test "$f"
xfortran     240    0                    then
xfortran     241    0                            if /bin/test "$w"
xfortran     242    0                            then
xfortran     243    0                                    f77 -$c -$f -$w $x
xfortran     244    0                            else
xfortran     245    0                                    f77 -$c -$f $x
xfortran     246    0                            fi
xfortran     247    0                    else
xfortran     248    0                            if /bin/test "$w"
xfortran     249    0                            then
xfortran     250    0                                    f77 -$c -$w $x
```

```
file        line level  source

xfortran     251    0                       else
xfortran     252    0                               f77 -$c $x
xfortran     253    0                       fi
xfortran     254    0               fi
xfortran     255    0       else
xfortran     256    0               if /bin/test "$f"
xfortran     257    0               then
xfortran     258    0                       if /bin/test "$w
xfortran     259    0                       then
xfortran     260    0                               f77 -$f -$w $x
xfortran     261    0                       else
xfortran     262    0                               f77 -$f $x
xfortran     263    0                       fi
xfortran     264    0               else
xfortran     265    0                       if /bin/test "$w'
xfortran     266    0                       then
xfortran     267    0                               f77 -$w $x
xfortran     268    0                       else
xfortran     269    0                               f77 $x
xfortran     270    0                       fi
xfortran     271    0               fi
xfortran     272    0       fi
xfortran     273    0       #
xfortran     274    0       #  Echo command to execute program and execute program
xfortran     275    0       #
xfortran     276    0
xfortran     277    0       if /bin/test "$t" = "$yes'
xfortran     278    0       then
xfortran     279    0               echo -n "!!!a.out '
xfortran     280    0               if /bin/test "$u" = "$yes"
xfortran     281    0               then
xfortran     282    0                       echo -n '< $v '
xfortran     283    0               fi
xfortran     284    0               if /bin/test "$z' = "$yes"
xfortran     285    0               then
xfortran     286    0                       echo -n "> $y'
xfortran     287    0               fi
xfortran     288    0               echo " "
xfortran     289    0               if /bin/test "$u' = "$yes'
xfortran     290    0               then
xfortran     291    0                       if /bin/test "$z  = "$yes"
xfortran     292    0                       then
xfortran     293    0                               a.out < $v > $y
xfortran     294    0                       else
xfortran     295    0                               a.out < $v
xfortran     296    0                       fi
xfortran     297    0               else
xfortran     298    0                       if /bin/test "$z  = "$yes
xfortran     299    0                       then
xfortran     300    0                               a.out > $y
```

```
file        line level  source

xfortran     301    0                          else
xfortran     302    0                                  a.out
xfortran     303    0                          fi
xfortran     304    0                fi
xfortran     305    0
xfortran     306    0       fi
xfortran     307    0
xfortran     308    0       echo '
xfortran     309    0            '
xfortran     310    0       echo -n "Press return to continue  "
xfortran     311    0       read y
xfortran     312    0
xfortran     313    0       exit 0
```

```
file          line level  source

xintroduction    1     0
xintroduction    2     0      # xintroduction
xintroduction    3     0      #
xintroduction    4     0      #       Purpose - to display a file which contains an introduction
xintroduction    5     0      #                 the system
xintroduction    6     0      #
xintroduction    7     0      #       Method -  A file called introduction must exist.  This file
xintroduction    8     0      #                 is displayed using the more command, thus the contents
xintroduction    9     0      #                 of the file is displayed one screen at a time.
xintroduction   10     0      #                 The file introduction can be easily modified to
xintroduction   11     0      #                 reflect changes to the system.
xintroduction   12     0      #
xintroduction   13     0      echo '
xintroduction   14     0
xintroduction   15     0                           MENU SHELL INTRODUCTION
xintroduction   16     0           "
xintroduction   17     0      cat introduction | more -d
xintroduction   18     0
xintroduction   19     0      echo -n "Press return to continue "
xintroduction   20     0      read x
xintroduction   21     0
xintroduction   22     0      exit 0
```

```
file            line level  source

xlistdirfiles     51     0                        1)  echo -n "*Do you wish a brief or long listing (b or l)?  ";;
xlistdirfiles     52     0                        2)  echo -n "*(b or l)?  ";;
xlistdirfiles     53     0                esac
xlistdirfiles     54     0
xlistdirfiles     55     0                read y
xlistdirfiles     56     0                if /bin/test "$y" = "$question"
xlistdirfiles     57     0                then
xlistdirfiles     58     0                        echo " "
xlistdirfiles     59     0                        echo "      A brief listing will only give the names of the files whereas a long"
xlistdirfiles     60     0                        echo "listing gives the mode, number of links, owner, size in bytes and time of"
xlistdirfiles     61     0                        echo "last modification in addition to the file names."
xlistdirfiles     62     0                fi
xlistdirfiles     63     0        done
xlistdirfiles     64     0
xlistdirfiles     65     0        while /bin/test "$z"
xlistdirfiles     66     0        do
xlistdirfiles     67     0                echo "
xlistdirfiles     68     0                "
xlistdirfiles     69     0                case $y in
xlistdirfiles     70     0                        b)  echo "!!! l $x
xlistdirfiles     71     0                            ";
xlistdirfiles     72     0                            l $x;
xlistdirfiles     73     0                            z= ;;
xlistdirfiles     74     0                        l)  echo "!!! ls -l $x
xlistdirfiles     75     0                            ";
xlistdirfiles     76     0                            ls -l $x;
xlistdirfiles     77     0                            z= ;;
xlistdirfiles     78     0                        *)  echo "      You must enter \"b\  for a brief listing or \"l\  for a long listing
xlistdirfiles     79     0                            echo -n "(b or l)?  ";
xlistdirfiles     80     0                            read y;;
xlistdirfiles     81     0                esac
xlistdirfiles     82     0        done
xlistdirfiles     83     0
xlistdirfiles     84     0        echo '
xlistdirfiles     85     0        '
xlistdirfiles     86     0        echo -n "Press return to continue  "
xlistdirfiles     87     0        read z
xlistdirfiles     88     0
xlistdirfiles     89     0        exit 0
```

```
file           line level  source

xlistdirfiles     1     0
xlistdirfiles     2     0       #xlistdirfiles
xlistdirfiles     3     0       #
xlistdirfiles     4     0       #       Purpose - To provide a means for displaying the files in a
xlistdirfiles     5     0       #                 directory.
xlistdirfiles     6     0       #
xlistdirfiles     7     0       #       Method -  The user must name the directory of which he
xlistdirfiles     8     0       #                 wishes to see the contents.  The uses must
xlistdirfiles     9     0       #                 also specify "b" or "l" for a brief or long
xlistdirfiles    10     0       #                 listing respectively.  If the user presses
xlistdirfiles    11     0       #                 return when first asked for the name of the
xlistdirfiles    12     0       #                 directory, the program will exit.
xlistdirfiles    13     0       #
xlistdirfiles    14     0       #       Error checking - The user is notified if he enters the
xlistdirfiles    15     0       #                 an invalid option.
xlistdirfiles    16     0       #
xlistdirfiles    17     0       question=?
xlistdirfiles    18     0       x=?
xlistdirfiles    19     0       y=?
xlistdirfiles    20     0       z=?
xlistdirfiles    21     0
xlistdirfiles    22     0       echo '
xlistdirfiles    23     0
xlistdirfiles    24     0
xlistdirfiles    25     0                               LIST CONTENTS OF A DIRECTORY
xlistdirfiles    26     0
xlistdirfiles    27     0
xlistdirfiles    28     0       while /bin/test "$x" = "$question"
xlistdirfiles    29     0       do
xlistdirfiles    30     0               echo ' '
xlistdirfiles    31     0               case $1 in
xlistdirfiles    32     0                       1)  echo -n "*What is the name of the directory of which you wish to see the content
xlistdirfiles    33     0                       2)  echo -n "*Directory name?  ";;
xlistdirfiles    34     0               esac
xlistdirfiles    35     0
xlistdirfiles    36     0               read x
xlistdirfiles    37     0
xlistdirfiles    38     0               if /bin/test "$x  = "$question"
xlistdirfiles    39     0               then
xlistdirfiles    40     0                       echo " "
xlistdirfiles    41     0                       echo "      You may see the contents of any directory in your account.  Enter"
xlistdirfiles    42     0                       echo "the full path name of the directory whose contents you wish to see.  If"
xlistdirfiles    43     0                       echo "you wish to see the contents of the current directory simply press return."
xlistdirfiles    44     0               fi
xlistdirfiles    45     0       done
xlistdirfiles    46     0
xlistdirfiles    47     0       while /bin/test "$y  = "$question"
xlistdirfiles    48     0       do
xlistdirfiles    49     0               echo '
xlistdirfiles    50     0               case $1 in
```

```
file        line level  source

xlistfiles     1     0
xlistfiles     2     0       # xlistfiles
xlistfiles     3     0       #
xlistfiles     4     0       #       Purpose - To provide a means for listing the files in the
xlistfiles     5     0       #                 current directory.
xlistfiles     6     0       #
xlistfiles     7     0       #       Method -  The user must specify a brief or long listing of
xlistfiles     8     0       #                 files.
xlistfiles     9     0       #
xlistfiles    10     0       #       Error checking - The user must specify "b  or "l" for a
xlistfiles    11     0       #                 brief or long listing respectively.  The user is
xlistfiles    12     0       #                 notified of an invalid option.
xlistfiles    13     0       #
xlistfiles    14     0       question=?
xlistfiles    15     0       x=?
xlistfiles    16     0       y=?
xlistfiles    17     0
xlistfiles    18     0       echo '
xlistfiles    19     0
xlistfiles    20     0                                   LIST NAMES OF FILES
xlistfiles    21     0
xlistfiles    22     0            '
xlistfiles    23     0
xlistfiles    24     0       while /bin/test "$x" = "$question'
xlistfiles    25     0       do
xlistfiles    26     0               echo ' '
xlistfiles    27     0               case $1 in
xlistfiles    28     0                       1)  echo -n "*Do you wish a brief or long listing (b or l)?  ";;
xlistfiles    29     0                       2)  echo -n "*(b or l)?  ";;
xlistfiles    30     0               esac
xlistfiles    31     0
xlistfiles    32     0               read x
xlistfiles    33     0
xlistfiles    34     0               if /bin/test "$x'
xlistfiles    35     0               then
xlistfiles    36     0                       if /bin/test "$x' = "$question
xlistfiles    37     0                       then
xlistfiles    38     0                               echo '
xlistfiles    39     0                               echo '      A brief listing will only give the names of the files whereas a l
xlistfiles    40     0                               echo "listing gives the mode, number of links, owner, size in bytes and time
xlistfiles    41     0                               echo "last modification in addition to the file names."
xlistfiles    42     0                       fi
xlistfiles    43     0               else
xlistfiles    44     0                       exit 0
xlistfiles    45     0               fi
xlistfiles    46     0       done
xlistfiles    47     0
xlistfiles    48     0       while /bin/test  $y' = "$question
xlistfiles    49     0       do
xlistfiles    50     0               echo
```

```
file          line level  source

xlistfiles      51     0
xlistfiles      52     0                case $x in
xlistfiles      53     0                        b)  echo "!!! l
xlistfiles      54     0                            ";
xlistfiles      55     0                            l;
xlistfiles      56     0                            y=0;;
xlistfiles      57     0                        l)  echo "!!! ls -l
xlistfiles      58     0                            ";
xlistfiles      59     0                            ls -l;
xlistfiles      60     0                            y=0;;
xlistfiles      61     0                        *)  echo "You must enter \"b\"  for a brief listing or \"l\" for a long listing";
xlistfiles      62     0                            echo -n "(b or l)?  ";
xlistfiles      63     0                            read x;;
xlistfiles      64     0                esac
xlistfiles      65     0        done
xlistfiles      66     0
xlistfiles      67     0        echo '
xlistfiles      68     0
xlistfiles      69     0        echo -n "Press return to continue  '
xlistfiles      70     0        read z
xlistfiles      71     0
xlistfiles      72     0        exit 0
```

```
file            line level  source

xmanual            1     0
xmanual            2     0        #xmanual
xmanual            3     0        #
xmanual            4     0        #       Purpose - To provide a means of displaying a command
xmanual            5     0        #                 description found in the on-line manual.
xmanual            6     0        #
xmanual            7     0        #       Method -  The user must name the command he wishes to
xmanual            8     0        #                 have displayed.  If he presses return when
xmanual            9     0        #                 first asked for the command, the program will
xmanual           10     0        #                 exit.
xmanual           11     0        #
xmanual           12     0        question=?
xmanual           13     0        x=?
xmanual           14     0
xmanual           15     0        echo '
xmanual           16     0
xmanual           17     0                                DISPLAY UNIX MANUAL
xmanual           18     0
xmanual           19     0             The \"man\" command is used to display the appropriate section of
xmanual           20     0        the UNIX manual for the UNIX command you specify.
xmanual           21     0             '
xmanual           22     0
xmanual           23     0
xmanual           24     0        while /bin/test "$x" = "$question
xmanual           25     0        do
xmanual           26     0                echo ' "
xmanual           27     0                case $1 in
xmanual           28     0                        1)  echo -n "*What UNIX command do you wish to see?  ";;
xmanual           29     0                        2)  echo -n "*UNIX command?  ";;
xmanual           30     0                esac
xmanual           31     0
xmanual           32     0                read x
xmanual           33     0
xmanual           34     0                if /bin/test "$x"
xmanual           35     0                then
xmanual           36     0                        if /bin/test "$x" = "$question
xmanual           37     0                        then
xmanual           38     0                                echo '
xmanual           39     0                                echo '      You must enter a UNIX command exactly as it is used.  For'
xmanual           40     0                                echo 'example if you wish to learn about the copy command enter \"cp\"'"
xmanual           41     0                        fi
xmanual           42     0                else
xmanual           43     0                        exit 0
xmanual           44     0                fi
xmanual           45     0        done
xmanual           46     0
xmanual           47     0        echo '
xmanual           48     0        !!! man $x
xmanual           49     0
xmanual           50     0
```

```
file            line level  source

xmanual           51     0       man $x
xmanual           52     0
xmanual           53     0       echo '
xmanual           54     0
xmanual           55     0       echo -n "Press return to continue
xmanual           56     0
xmanual           57     0       read z
xmanual           58     0
xmanual           59     0       exit 0
```

```
file          line level  source

xmorefile        1     0
xmorefile        2     0       # xmorefile
xmorefile        3     0       #
xmorefile        4     0       #       Purpose - To provide a means for displaying a file one screen
xmorefile        5     0       #                 at a time.
xmorefile        6     0       #
xmorefile        7     0       #       Method -  The user must enter the name of the file he wishes
xmorefile        8     0       #                 to have displayed.  If the user presses return when
xmorefile        9     0       #                 he is first asked for the name of the file, the
xmorefile       10     0       #                 program will exit.
xmorefile       11     0       #
xmorefile       12     0       #       Error checking - If the user enters the name of a nonexistent
xmorefile       13     0       #                 file, the program tells the user the file does not
xmorefile       14     0       #                 exist and then exits.
xmorefile       15     0       #
xmorefile       16     0       question=?
xmorefile       17     0       x=?
xmorefile       18     0
xmorefile       19     0       echo "
xmorefile       20     0
xmorefile       21     0                       DISPLAY ONE SCREEN OF TEXT AT A TIME
xmorefile       22     0
xmorefile       23     0
xmorefile       24     0
xmorefile       25     0       while /bin/test "$x" = "$question"
xmorefile       26     0       do
xmorefile       27     0               echo " .
xmorefile       28     0               case $1 in
xmorefile       29     0                       1)  echo -n "*What is the name of the file you wish to display?  ";;
xmorefile       30     0                       2)  echo -n "*File name?   ";;
xmorefile       31     0               esac
xmorefile       32     0
xmorefile       33     0               read x
xmorefile       34     0
xmorefile       35     0               if /bin/test "$x"
xmorefile       36     0               then
xmorefile       37     0                       if /bin/test "$x" = "$question"
xmorefile       38     0                       then
xmorefile       39     0                               echo " .
xmorefile       40     0                               echo "      You must enter the name of an existing file.  The screen movement
xmorefile       41     0                               echo "will stop when a complete screen has been displayed.  To restart the"
xmorefile       42     0                               echo "press the \"space bar.\""
xmorefile       43     0                       fi
xmorefile       44     0               else
xmorefile       45     0                       exit 0
xmorefile       46     0               fi
xmorefile       47     0       done
xmorefile       48     0
xmorefile       49     0       if /bin/test -f "$x"
xmorefile       50     0       then
```

```
file          line level  source

xmorefile       51     0              echo '
xmorefile       52     0              '
xmorefile       53     0              echo "!!! more -d $x"
xmorefile       54     0              echo "
xmorefile       55     0              '
xmorefile       56     0
xmorefile       57     0              more -d $x
xmorefile       58     0
xmorefile       59     0              echo "
xmorefile       60     0                   "
xmorefile       61     0              echo -n "Press return to continue  '
xmorefile       62     0              read z
xmorefile       63     0       else
xmorefile       64     0              echo '  '
xmorefile       65     0              echo "File $x does not exist!"
xmorefile       66     0              sleep 2
xmorefile       67     0       fi
xmorefile       68     0
xmorefile       69     0       exit 0
```

```
file            line level  source

xmovefile          1     0
xmovefile          2     0       # xmovefile
xmovefile          3     0       #
xmovefile          4     0       #       Purpose - To provide a means for moving a file from one
xmovefile          5     0       #                 name to another.
xmovefile          6     0       #
xmovefile          7     0       #       Method -  The user is asked for the name of the file which
xmovefile          8     0       #                 he wishes to move and the name of the file where
xmovefile          9     0       #                 he wishes it moved to.  If the user presses return
xmovefile         10     0       #                 as the source file when first asked the program
xmovefile         11     0       #                 will exit.
xmovefile         12     0       #
xmovefile         13     0       #       Error checking - If the user gives the name of a nonexistent
xmovefile         14     0       #                 file as the source file, he will be told the file
xmovefile         15     0       #                 does not exist and the program will exit.
xmovefile         16     0       #
xmovefile         17     0       question=?
xmovefile         18     0       x=?
xmovefile         19     0       y=?
xmovefile         20     0
xmovefile         21     0       echo '
xmovefile         22     0
xmovefile         23     0                 MOVE A FILE (CHANGE ITS NAME)
xmovefile         24     0             '
xmovefile         25     0
xmovefile         26     0       while /bin/test "$x" = "$question'
xmovefile         27     0       do
xmovefile         28     0               echo ' "
xmovefile         29     0
xmovefile         30     0               case $1 in
xmovefile         31     0                       1)  echo -n "*What is the name of the file you wish to move?  ";;
xmovefile         32     0                       2)  echo -n "*Old file?  ";;
xmovefile         33     0               esac
xmovefile         34     0
xmovefile         35     0               read x
xmovefile         36     0
xmovefile         37     0               if /bin/test "$x'
xmovefile         38     0               then
xmovefile         39     0
xmovefile         40     0                       if test "$x' = "$question
xmovefile         41     0                       then
xmovefile         42     0                               echo " "
xmovefile         43     0                               echo '      You must enter the name of an existing file.  If the file is not"
xmovefile         44     0                               echo "in the current directory the entire path must be entered."
xmovefile         45     0                       fi
xmovefile         46     0               else
xmovefile         47     0                       exit 0
xmovefile         48     0               fi
xmovefile         49     0       done
xmovefile         50     0
```

```
file        line level  source

xmovefile     51     0    if /bin/test -f $x
xmovefile     52     0    then
xmovefile     53     0            while test "$y" - "$question
xmovefile     54     0            do
xmovefile     55     0                    echo " "
xmovefile     56     0                    case $1 in
xmovefile     57     0                            1)  echo -n "*What is the name of the file where you wish to move $x?  ";;
xmovefile     58     0                            2)  echo -n "*New file?  ";;
xmovefile     59     0                    esac
xmovefile     60     0
xmovefile     61     0                    read y
xmovefile     62     0
xmovefile     63     0                    if /bin/test "$y
xmovefile     64     0                    then
xmovefile     65     0                            if test "$y" = "$question
xmovefile     66     0                            then
xmovefile     67     0                                    echo " "
xmovefile     68     0                                    echo "     You must enter the new name for $x.  The contents of $x
xmovefile     69     0                                    echo "will be moved to the file name you enter.  If you enter the na
xmovefile     70     0                                    echo "existing file the contents of that file will be lost.  The fil
xmovefile     71     0                                    echo "replaced with the contents of $x."
xmovefile     72     0                            fi
xmovefile     73     0                    else
xmovefile     74     0                            echo "
xmovefile     75     0                                    "
xmovefile     76     0                            echo "     You must enter the file name where $x will be moved.
xmovefile     77     0                            y=?
xmovefile     78     0                    fi
xmovefile     79     0            done
xmovefile     80     0
xmovefile     81     0            echo "
xmovefile     82     0                    "
xmovefile     83     0            echo "!!! mv $x $y"
xmovefile     84     0
xmovefile     85     0            mv $x $y
xmovefile     86     0
xmovefile     87     0            echo "
xmovefile     88     0                    "
xmovefile     89     0            echo -n "Press return to continue  "
xmovefile     90     0
xmovefile     91     0            read z
xmovefile     92     0
xmovefile     93     0    else
xmovefile     94     0            echo    "
xmovefile     95     0            echo "File $x does not exist!"
xmovefile     96     0            sleep 2
xmovefile     97     0    fi
xmovefile     98     0
xmovefile     99     0    exit 0
```

```
file          line level  source

xoctalfile       1     0
xoctalfile       2     0       # xoctalfile
xoctalfile       3     0       #
xoctalfile       4     0       #       Purpose - To provide a means for displaying a file in octal,
xoctalfile       5     0       #                 ASCII, decimal, or hexadecimal form.
xoctalfile       6     0       #
xoctalfile       7     0       #       Method -  The user must enter the name of the file he wishes
xoctalfile       8     0       #                 have displayed.  If the user presses return when first
xoctalfile       9     0       #                 asked for the file name, the program will exit.
xoctalfile      10     0       #                 The user must specify the form of display he desires
xoctalfile      11     0       #                 from the list of options he is given.
xoctalfile      12     0       #
xoctalfile      13     0       #       Error checking - If the user enters a nonexistent file,
xoctalfile      14     0       #                 the program will tell him the file does exist
xoctalfile      15     0       #                 and then exit.  The program verifies that the
xoctalfile      16     0       #                 user selects a valid format for display.
xoctalfile      17     0       #
xoctalfile      18     0       question=?
xoctalfile      19     0       x=?
xoctalfile      20     0       y=?
xoctalfile      21     0
xoctalfile      22     0       echo "
xoctalfile      23     0
xoctalfile      24     0                               OCTAL DISPLAY OF A FILE
xoctalfile      25     0
xoctalfile      26     0
xoctalfile      27     0
xoctalfile      28     0       while /bin/test "$x" = "$question"
xoctalfile      29     0       do
xoctalfile      30     0               echo " "
xoctalfile      31     0               case $1 in
xoctalfile      32     0                       1)  echo -n "*What is the name of the file you wish to display?  ";;
xoctalfile      33     0                       2)  echo -n "*File name?  ";;
xoctalfile      34     0               esac
xoctalfile      35     0
xoctalfile      36     0               read x
xoctalfile      37     0
xoctalfile      38     0               if /bin/test "$x"
xoctalfile      39     0               then
xoctalfile      40     0                       if /bin/test "$x" = "$question"
xoctalfile      41     0                       then
xoctalfile      42     0                               echo " "
xoctalfile      43     0                               echo "     You must enter the name of an existing file.  If the file is not"
xoctalfile      44     0                               echo "in the current directory the entire path name must be entered."
xoctalfile      45     0                       fi
xoctalfile      46     0               else
xoctalfile      47     0                       exit 0
xoctalfile      48     0               fi
xoctalfile      49     0       done
xoctalfile      50     0
```

```
file          line level  source

xoctalfile      51     0      if /bin/test -f $x
xoctalfile      52     0      then
xoctalfile      53     0              while /bin/test "$y" = "$question"
xoctalfile      54     0              do
xoctalfile      55     0                      echo " "
xoctalfile      56     0                      echo "Octal displays may be done is one of the following formats:"
xoctalfile      57     0                      echo "      b)  Interpret bytes in octal"
xoctalfile      58     0                      echo "      c)  Interpret bytes in ASCII"
xoctalfile      59     0                      echo "      d)  Interpret words in decimal"
xoctalfile      60     0                      echo "      o)  Interpret words in octal"
xoctalfile      61     0                      echo "      x)  Interpret words in hex"
xoctalfile      62     0                      echo "
xoctalfile      63     0
xoctalfile      64     0                      echo -n "Enter \"b, c, d, o, x\" indicating the format you wish.  "
xoctalfile      65     0                      read y
xoctalfile      66     0
xoctalfile      67     0                      echo "
xoctalfile      68     0
xoctalfile      69     0                      case $y in
xoctalfile      70     0                              b|c|d|o|x) echo "!!! od -$y $x" ;;
xoctalfile      71     0                              *)  y=?;;
xoctalfile      72     0                      esac
xoctalfile      73     0              done
xoctalfile      74     0
xoctalfile      75     0              echo "
xoctalfile      76     0
xoctalfile      77     0
xoctalfile      78     0              od -$y $x
xoctalfile      79     0
xoctalfile      80     0              echo "
xoctalfile      81     0                    "
xoctalfile      82     0              echo -n "Press return to continue  "
xoctalfile      83     0              read z
xoctalfile      84     0
xoctalfile      85     0      else
xoctalfile      86     0              echo " "
xoctalfile      87     0              echo "File $x does not exist!"
xoctalfile      88     0      fi
xoctalfile      89     0
xoctalfile      90     0      exit 0
```

```
file          line level  source

xpascal          1     0
xpascal          2     0       #xpascal
xpascal          3     0       #
xpascal          4     0       #       Purpose - To provide a means to compile and execute a
xpascal          5     0       #                 Pascal program.
xpascal          6     0       #
xpascal          7     0       #       Method -  The user the name the file he wishes to compile.
xpascal          8     0       #                 If the user presses return when first asked the
xpascal          9     0       #                 file name, the program will exit.  The user is given
xpascal         10     0       #                 the option for an input file and listing file.
xpascal         11     0       #
xpascal         12     0       #       Error checking -  If the user names a nonexistent input
xpascal         13     0       #                 file or a nonexistent pascal source file, the user
xpascal         14     0       #                 will be notified that the file does not exist.
xpascal         15     0       #                 The program will then exit.
xpascal         16     0       #
xpascal         17     0       question=?
xpascal         18     0       yes=y
xpascal         19     0       t=?             #listing flag
xpascal         20     0       u=?             # input file flag
xpascal         21     0       v=?             # input file name
xpascal         22     0       x=?             # source file name
xpascal         23     0       z=?             # listing in a file flag
xpascal         24     0       w=?             # listing file name
xpascal         25     0
xpascal         26     0       echo
xpascal         27     0
xpascal         28     0                                   PASCAL PROGRAM
xpascal         29     0
xpascal         30     0       You should expect a delay when your program is compiled and executed.
xpascal         31     0       Program compilation requires more time then most of the tasks found in
xpascal         32     0       the Menu shell.
xpascal         33     0
xpascal         34     0
xpascal         35     0
xpascal         36     0
xpascal         37     0       #
xpascal         38     0       #  Request source file
xpascal         39     0       #
xpascal         40     0       while /bin/test "$x" = "$question
xpascal         41     0       do
xpascal         42     0               echo
xpascal         43     0               case $1 in
xpascal         44     0                       1)  echo -n "*What is the name of PASCAL source file you wish to run?  ";;
xpascal         45     0                       2)  echo -n "*Source file?  ";;
xpascal         46     0               esac
xpascal         47     0
xpascal         48     0               read x
xpascal         49     0
xpascal         50     0               if /bin/test "$x"
```

```
file            line level  source

xpascal           51     0                then
xpascal           52     0                        if /bin/test "$x" = "$question"
xpascal           53     0                        then
xpascal           54     0                                echo '
xpascal           55     0                                echo '      You must enter the name of a file which contains a PASCAL"
xpascal           56     0                                echo "program.  The file name must end in \".p\""
xpascal           57     0                        fi
xpascal           58     0                else
xpascal           59     0                        exit 0
xpascal           60     0                fi
xpascal           61     0        done
xpascal           62     0
xpascal           63     0        #
xpascal           64     0        #  Request input file
xpascal           65     0        #
xpascal           66     0        if /bin/test -f "$x"
xpascal           67     0        then
xpascal           68     0                while /bin/test "$u  = "$question"
xpascal           69     0                do
xpascal           70     0                        echo '
xpascal           71     0                        case $1 in
xpascal           72     0                                1) echo -n "*Do you have an input file (y or n)?  ";;
xpascal           73     0                                2) echo -n '*Input file (y or n)?  ';;
xpascal           74     0                        esac
xpascal           75     0
xpascal           76     0                        read u
xpascal           77     0
xpascal           78     0                        case $u in
xpascal           79     0                                y | Y)  echo ' ';
xpascal           80     0                                        echo -n 'What is the name of the input file?  ';
xpascal           81     0                                        read v;
xpascal           82     0                                        if  /bin/test -f  "$v"
xpascal           83     0                                        then
xpascal           84     0                                                echo
xpascal           85     0                                        else
xpascal           86     0                                                echo
xpascal           87     0                                                echo 'File $v does not exist!  '
xpascal           88     0                                                exit 0
xpascal           89     0                                        fi;;
xpascal           90     0                                n | N)  u= ;;
xpascal           91     0                                \?)     echo " ";
xpascal           92     0                                        echo "      A program which expects data to be input from the termina
xpascal           93     0                                        echo "the data redirected from an input file.  If your program is ex
xpascal           94     0                                        echo "data from standard input and the actual data is in a file answ
xpascal           95     0                                        echo "to this question.";;
xpascal           96     0                                *)      u=?;
xpascal           97     0                                        echo    ;;
xpascal           98     0                        esac
xpascal           99     0                done
xpascal          100     0
```

```
file          line level  source

xpascal        101     0       #
xpascal        102     0       #       Determine if user desires a listing
xpascal        103     0       #
xpascal        104     0               while /bin/test "$t  = "$question
xpascal        105     0               do
xpascal        106     0                       echo '
xpascal        107     0                       case $1 in
xpascal        108     0                               1)  echo -n "*Do you want a listing of your program (y or n)?  ";;
xpascal        109     0                               2)  echo -n "*Listing of program (y or n)?  ";;
xpascal        110     0                       esac
xpascal        111     0
xpascal        112     0                       read t
xpascal        113     0
xpascal        114     0                       case $t in
xpascal        115     0                               y | Y) y=y;;
xpascal        116     0       #
xpascal        117     0       #                       Compile and execute program with no listing
xpascal        118     0       #
xpascal        119     0                               n | N) echo    ";
xpascal        120     0                                       if /bin/test "$u
xpascal        121     0                                       then
xpascal        122     0                                               echo "!!! pix $x < $v"
xpascal        123     0                                               pix $x < $v
xpascal        124     0                                       else
xpascal        125     0                                               echo "!!! pix $x
xpascal        126     0                                               pix $x
xpascal        127     0                                       fi;
xpascal        128     0                                       w=";;
xpascal        129     0                               \?) echo "";
xpascal        130     0                                  echo "     Any errors in the source file will be printed out, however";
xpascal        131     0                                   echo "to get a complete listing of your program you must request one.";
xpascal        132     0                                   echo "  ";;
xpascal        133     0                               *) t=?;
xpascal        134     0                                  echo "  ";;
xpascal        135     0                       esac
xpascal        136     0               done
xpascal        137     0       else
xpascal        138     0               echo "File $x does not exit!"
xpascal        139     0               exit 0
xpascal        140     0
xpascal        141     0       fi
xpascal        142     0
xpascal        143     0       #
xpascal        144     0       #  Determine if user wants listing in a file
xpascal        145     0       #
xpascal        146     0       if /bin/test  $y  = "$yes
xpascal        147     0       then
xpascal        148     0               while /bin/test "$z  = "$question"
xpascal        149     0               do
xpascal        150     0                       echo
```

```
file          line level  source

xpascal        151     0                        echo -n "*Do you want the listing printed to a file (y or n)?  "
xpascal        152     0                        read z
xpascal        153     0
xpascal        154     0                        case $z in
xpascal        155     0                                y | Y) z=y;;
xpascal        156     0       #
xpascal        157     0       #                        Compile and execute programs with listing at terminal
xpascal        158     0       #
xpascal        159     0                                n | N) echo    ";
xpascal        160     0                                       if /bin/test "$u
xpascal        161     0                                       then
xpascal        162     0                                               echo "!!! pic -l $x < $v"
xpascal        163     0                                               pix -l $x < $v
xpascal        164     0                                       else
xpascal        165     0                                               echo "!!! pic -l $x"
xpascal        166     0                                               pix -l $x
xpascal        167     0                                       fi;
xpascal        168     0                                       w= ;;
xpascal        169     0                                \?)  echo "    ";
xpascal        170     0                                     echo "       The listing will be printed at the terminal unless you";
xpascal        171     0                                    echo "you request the listing to be placed in a file.  The program";
xpascal        172     0                                    echo "output will also be placed in the file.";;
xpascal        173     0                                *) z=?;
xpascal        174     0                                   echo    ";;
xpascal        175     0                        esac
xpascal        176     0                done
xpascal        177     0       fi
xpascal        178     0
xpascal        179     0       #
xpascal        180     0       #  Request listing file name
xpascal        181     0       #
xpascal        182     0       if /bin/test "$z  = "$yes
xpascal        183     0       then
xpascal        184     0               while /bin/test "$w" = "$question"
xpascal        185     0               do
xpascal        186     0                       echo "
xpascal        187     0                       case $1 in
xpascal        188     0                               1)  echo -n "*What is the name of the listing file?  ";;
xpascal        189     0                               2)  echo -n "*listing file?  ";;
xpascal        190     0                       esac
xpascal        191     0
xpascal        192     0                       read w
xpascal        193     0
xpascal        194     0                       case $w in
xpascal        195     0                               \?)  echo    ;
xpascal        196     0                                    echo "       You must enter a new file name.  If you enter the name of an
xpascal        197     0                                    echo "file the contents of that file will be lost.  The file will be re
xpascal        198     0                                    echo "with the contents of the listing file.";;
xpascal        199     0                               *)  if /bin/test "$w"
xpascal        200     0                                   then
```

```
file            line level  source

xpascal          201     0                                          echo
xpascal          202     0                                      else
xpascal          203     0                                          w=?
xpascal          204     0                                          echo " '
xpascal          205     0                                      fi;;
xpascal          206     0                              esac
xpascal          207     0                      done
xpascal          208     0          fi
xpascal          209     0
xpascal          210     0          #
xpascal          211     0          #  Compile and execute program with listing file
xpascal          212     0          #
xpascal          213     0          if /bin/test  $w"
xpascal          214     0          then
xpascal          215     0                              echo " '
xpascal          216     0                              if /bin/test "$u
xpascal          217     0                              then
xpascal          218     0                                      echo "!!!pix -l $x < $v > $w
xpascal          219     0                                      pix -l $x < $v > $w
xpascal          220     0                              else
xpascal          221     0                                      echo "!!! pix -l $x > $w'
xpascal          222     0                                      pix -l $x > $w
xpascal          223     0                              fi
xpascal          224     0          fi
xpascal          225     0          echo
xpascal          226     0
xpascal          227     0          echo -n "Press return to continue  '
xpascal          228     0          read z
xpascal          229     0
xpascal          230     0          exit 0
```

```
file        line level  source

xpasswcrd      1     0
xpasswcrd      2     0      #xpassword
xpasswcrd      3     0      #
xpasswcrd      4     0      #       Purpose - To provide a means to change the users password.
xpasswcrd      5     0      #
xpasswcrd      6     0      #       Method -  A brief explanation of the "passwd" command is
xpasswcrd      7     0      #                 given.  The shell then issues the "passwd" command.
xpasswcrd      8     0      #
xpasswcrd      9     0      echo '
xpassword     10     0
xpasswcrd     11     0                                CHANGE PASSWORD
xpasswcrd     12     0
xpassword     13     0
xpasswcrd     14     0           You may change your password by using the \"passwd\' command.
xpassword     15     0      \"Passwd\" will ask you for your current password and then ask for
xpassword     16     0      your new password two times.  If your new password is less than 6
xpasswcrd     17     0      characters in length you will be asked for a longer password since
xpassword     18     0      a longer password is generally more secure; however if you persist
xpasswcrd     19     0      a short password will be accepted.
xpasswcrd     20     0
xpassword     21     0
xpasswcrd     22     0      !!! passwd
xpasswcrd     23     0           '
xpasswcrd     24     0
xpasswcrd     25     0
xpassword     26     0      passwd
xpasswcrd     27     0
xpasswcrd     28     0      echo
xpasswcrd     29     0      echo -n "Press return to continue
xpasswcrd     30     0      read z
xpasswcrd     31     0
xpasswcrd     32     0      exit 0
```

```
file          line level  source

xpathident       1     0
xpathident       2     0       #xpathident
xpathident       3     0       #
xpathident       4     0       #       Purpose - To provide a means of printing the current path.
xpathident       5     0       #
xpathident       6     0       #       Method -  A brief explanation of the pwd command is given.
xpathident       7     0       #                 The shell then issues the "pwd" command.
xpathident       8     0       #
xpathident       9     0       question=?
xpathident      10     0       x=?
xpathident      11     0
xpathident      12     0       echo '
xpathident      13     0
xpathident      14     0                                PATH IDENTIFICATION
xpathident      15     0
xpathident      16     0            The UNIX operating system uses a tree structure of directories.
xpathident      17     0       It is possible to see the ancestors of your current directory by
xpathident      18     0       using the \"pwd\" command.  The \"pwd\" command returns the full path
xpathident      19     0       name of your current working directory.
xpathident      20     0
xpathident      21     0
xpathident      22     0       !!! pwd
xpathident      23     0           '
xpathident      24     0
xpathident      25     0
xpathident      26     0       pwd
xpathident      27     0
xpathident      28     0       echo '
xpathident      29     0       '
xpathident      30     0       echo -n "Press return to continue
xpathident      31     0       read z
xpathident      32     0
xpathident      33     0       exit 0
```

```
file          line level  source

xprirtfile       1     0
xprintfile       2     0      # xprintfile
xprirtfile       3     0      #
xprintfile       4     0      #       Purpose - To provide a means of printing a file at a printer.
xprintfile       5     0      #
xprintfile       6     0      #       Method -  The user must enter the name of the file he wishes
xprintfile       7     0      #                 to have printed.  If the user presses return when
xprintfile       8     0      #                 first asked the name of the file the program will
xprintfile       9     0      #                 exit.  The user is given the option to have the file
xprintfile      10     0      #                 printed with line numbers.
xprirtfile      11     0      #
xprintfile      12     0      #       Error checking - If the user enters the name of a nonexistent
xprintfile      13     0      #                 file, the program tells the user that the file does
xprirtfile      14     0      #                 exist and exits.
xprintfile      15     0      #
xprirtfile      16     0      question=?
xprintfile      17     0      x=?
xprintfile      18     0
xprintfile      19     0      echo '
xprintfile      20     0
xprintfile      21     0                              PRINT A FILE AT PRINTER
xprirtfile      22     0
xprintfile      23     0           "
xprirtfile      24     0
xprirtfile      25     0      while /bin/test "$x" = '$question
xprirtfile      26     0      do
xprirtfile      27     0              echo    '
xprintfile      28     0              case $1 in
xprintfile      29     0                      1)  echo -n "*What is the name of the file you wish to print?  ";;
xprirtfile      30     0                      2)  echo -n "*File name?  ";;
xprirtfile      31     0              esac
xprintfile      32     0
xprintfile      33     0              read x
xprintfile      34     0
xprirtfile      35     0              if /bin/test "$x"
xprirtfile      36     0              then
xprintfile      37     0                      if /bin/test "$x" = "$question
xprintfile      38     0                      then
xprintfile      39     0                              echo " '
xprintfile      40     0                              echo "     You must enter the name of an existing file. This file will"
xprirtfile      41     0                              echo "be printed at the printer assigned to your system."
xprintfile      42     0                      fi
xprintfile      43     0              else
xprintfile      44     0                      exit 0
xprintfile      45     0              fi
xprintfile      46     0      done
xprirtfile      47     0
xprintfile      48     0      if /bin/test -f "$x'
xprirtfile      49     0      then
xprintfile      50     0              echo    "
```

```
file          line level  source

xprintfile      51     0              echo -n "Do you wish to have the line numbers printed (y or n)?  "
xprintfile      52     0
xprintfile      53     0              read y
xprintfile      54     0
xprintfile      55     0              echo '
xprintfile      56     0
xprintfile      57     0
xprintfile      58     0              case $y in
xprintfile      59     0                      y|Y)  echo "!!! cpr $x | lpr";
xprintfile      60     0                            echo "
xprintfile      61     0                                  ";
xprintfile      62     0                            echo "
xprintfile      63     0                                  YOUR FILE IS BEING PRINTED";
xprintfile      64     0                            echo " ";
xprintfile      65     0                            cpr $x | lpr;;
xprintfile      66     0
xprintfile      67     0                      *)  echo "!!! lpr $x";
xprintfile      68     0                          echo "
xprintfile      69     0                                ";
xprintfile      70     0                          echo "
xprintfile      71     0                                YOUR FILE IS BEING PRINTED";
xprintfile      72     0                          echo " ";
xprintfile      73     0                          lpr $x;;
xprintfile      74     0              esac
xprintfile      75     0              echo '
xprintfile      76     0
xprintfile      77     0              echo -n "Press return to continue
xprintfile      78     0              read z
xprintfile      79     0      else
xprintfile      80     0              echo ' '
xprintfile      81     0              echo "File $x does not exist!"
xprintfile      82     0              sleep 2
xprintfile      83     0      fi
xprintfile      84     0
xprintfile      85     0      exit 0
```

```
file           line level  source

xreceivemail      1     0
xreceivemail      2     0       #xreceivemail
xreceivemail      3     0       #
xreceivemail      4     0       #       Purpose - To provide a means for receiving mail.
xreceivemail      5     0       #
xreceivemail      6     0       #       Method -  A brief explanation of the "mail" command
xreceivemail      7     0       #                 is given.  If the prompt level is verbose,
xreceivemail      8     0       #                 a subset of the available mail options is
xreceivemail      9     0       #                 given.  The shell then issues the "mail"
xreceivemail     10     0       #                 command.
xreceivemail     11     0       #
xreceivemail     12     0       echo '
xreceivemail     13     0
xreceivemail     14     0                                   RECEIVE MAIL
xreceivemail     15     0
xreceivemail     16     0
xreceivemail     17     0            The \"mail\" command displays any mail you have received.  The
xreceivemail     18     0       messages are displayed in chronological order with the most recent
xreceivemail     19     0       one first.  After each message you will see a question mark.  If you
xreceivemail     20     0       type \"?\", return, you will see the valid responses to your mail.  You
xreceivemail     21     0       must respond to each message.
xreceivemail     22     0             '
xreceivemail     23     0
xreceivemail     24     0
xreceivemail     25     0       sleep 3
xreceivemail     26     0       if /bin/test $1 = 1
xreceivemail     27     0       then
xreceivemail     28     0               echo 'The following responses to mail are frequently used:'
xreceivemail     29     0               echo '   q          - quit reading your mail'
xreceivemail     30     0               echo '   w [file] - write the mail message to the file name you specify'
xreceivemail     31     0               echo '   +          - look at the next mail message (does not delete)'
xreceivemail     32     0               echo '   d          - delete current mail message'
xreceivemail     33     0       fi
xreceivemail     34     0
xreceivemail     35     0       sleep 7
xreceivemail     36     0
xreceivemail     37     0       echo "
xreceivemail     38     0             '
xreceivemail     39     0       echo "!!! mail"
xreceivemail     40     0
xreceivemail     41     0       mail
xreceivemail     42     0
xreceivemail     43     0       echo '
xreceivemail     44     0
xreceivemail     45     0       echo -n 'Press return to continue
xreceivemail     46     0       read z
xreceivemail     47     0
xreceivemail     48     0       exit 0
```

```
file        line level  source

xseefile       1     0
xseefile       2     0       # xseefile
xseefile       3     0       #
xseefile       4     0       #       Purpose - To provide a means for displaying a file with the
xseefile       5     0       #                 non-printing characters in a readable form.
xseefile       6     0       #
xseefile       7     0       #       Method -  The user must enter the name of the file he wishes
xseefile       8     0       #                 to have displayed.  If the user presses return
xseefile       9     0       #                 when first asked for the file name, the program
xseefile      10     0       #                 will exit.
xseefile      11     0       #
xseefile      12     0       #       Error checking - If the user enters the name of a nonexistent
xseefile      13     0       #                 file, the program will tell the user that the file
xseefile      14     0       #                 does not exist and exit.
xseefile      15     0       #
xseefile      16     0       question=?
xseefile      17     0       x=?
xseefile      18     0
xseefile      19     0       echo '
xseefile      20     0
xseefile      21     0                       DISPIAY NON-PRINTING CHARACTERS IN A FILE
xseefile      22     0
xseefile      23     0
xseefile      24     0       Your file will be displayed with the non-printing characters in a
xseefile      25     0       readable format.  The following formats are used:
xseefile      26     0            ^I = tat
xseefile      27     0            ^? = delete
xseefile      28     0            $  = end of line
xseefile      29     0
xseefile      30     0
xseefile      31     0       while /bin/test "$x" = "$question"
xseefile      32     0       do
xseefile      33     0               echo ' "
xseefile      34     0               case $1 in
xseefile      35     0                       1)  echo -n "*What is the name of the file you wish to display?  ";;
xseefile      36     0                       2)  echo -n "*File name?  ";;
xseefile      37     0               esac
xseefile      38     0
xseefile      39     0               read x
xseefile      40     0
xseefile      41     0               if /bin/test "$x'
xseefile      42     0               then
xseefile      43     0                       if /bin/test "$x" = "$question'
xseefile      44     0                       then
xseefile      45     0                               echo " '
xseefile      46     0                               echo '     You must enter the name of existing file.  If the file is not"
xseefile      47     0                               echo "in the current directory the entire path name must be entered."
xseefile      48     0                       fi
xseefile      49     0               else
xseefile      50     0                       exit 0
```

```
file        line level  source

xseefile      51     0                fi
xseefile      52     0        done
xseefile      53     0
xseefile      54     0        if /bin/test -f "$x"
xseefile      55     0        then
xseefile      56     0                echo '
xseefile      57     0
xseefile      58     0                echo "!!! see  -v -e -t $x"
xseefile      59     0                echo '
xseefile      60     0
xseefile      61     0
xseefile      62     0
xseefile      63     0                see -v -e -t  $x
xseefile      64     0
xseefile      65     0                echo '
xseefile      66     0                '
xseefile      67     0                echo -n "Press return to continue  '
xseefile      68     0                read z
xseefile      69     0        else
xseefile      70     0                echo ' "
xseefile      71     0                echo "File $x does not exist!"
xseefile      72     0                sleep 2
xseefile      73     0        fi
xseefile      74     0
xseefile      75     0        exit 0
```

```
file        line level  source

xsendmail      1     0
xsendmail      2     0       #xsendmail
xsendmail      3     0       #
xsendmail      4     0       #       Purpose - To provide a means for sending mail to another
xsendmail      5     0       #                 user.
xsendmail      6     0       #
xsendmail      7     0       #       Method -  The user must enter the login name of the user to
xsendmail      8     0       #                 to whom he wishes to send mail.  If the user presses
xsendmail      9     0       #                 return, the program will exit.  The user has the option
xsendmail     10     0       #                 of sending a previously existing file in the mail or
xsendmail     11     0       #                 sending a messages he composes at the terminal.
xsendmail     12     0       #
xsendmail     13     0       #       Error checking - If the user wants to send a nonexistent
xsendmail     14     0       #                 file, he is notified that the file does not
xsendmail     15     0       #                 exist.  The program then exits.
xsendmail     16     0       #
xsendmail     17     0       question=?
xsendmail     18     0       x=?
xsendmail     19     0       y=?
xsendmail     20     0
xsendmail     21     0
xsendmail     22     0
xsendmail     23     0       echo '
xsendmail     24     0
xsendmail     25     0                                       SEND MAIL
xsendmail     26     0
xsendmail     27     0            '
xsendmail     28     0
xsendmail     29     0       while /bin/test "$x" = "$question"
xsendmail     30     0       do
xsendmail     31     0               echo " "
xsendmail     32     0               case $1 in
xsendmail     33     0                       1)  echo -n "*To whom do you wish to send mail?  ";;
xsendmail     34     0                       2)  echo -n "*To whom?  ";;
xsendmail     35     0               esac
xsendmail     36     0
xsendmail     37     0               read x
xsendmail     38     0
xsendmail     39     0               if /bin/test "$x"
xsendmail     40     0               then
xsendmail     41     0                       if /bin/test "$x  = "$question"
xsendmail     42     0                       then
xsendmail     43     0                               echo ' '
xsendmail     44     0                               echo '      You must enter the login name of the person to whom you wish to"
xsendmail     45     0                               echo 'send mail."
xsendmail     46     0                       fi
xsendmail     47     0               else
xsendmail     48     0                       exit 0
xsendmail     49     0               fi
xsendmail     50     0       done
```

```
file          line level  source

xsendmail       51     0       while /bin/test "$y" = "$question"
xsendmail       52     0       do
xsendmail       53     0               echo "
xsendmail       54     0               case $1 in
xsendmail       55     0                       1)  echo -n "*Is the message you wish to send contained in a file (y or n)?  ";;
xsendmail       56     0                       2)  echo -n "*Send a file (y or n)?  ";;
xsendmail       57     0               esac
xsendmail       58     0
xsendmail       59     0               read y
xsendmail       60     0
xsendmail       61     0               case $y in
xsendmail       62     0                       \?)  echo " ";
xsendmail       63     0                            echo "You are able to send mail in two ways:";
xsendmail       64     0                            echo "  1)  Create a file whose contents is the message you wish to send.";
xsendmail       65     0                            echo "  2)  Enter the message to be sent at the terminal after the mail";
xsendmail       66     0                            echo "      command is initiated.";;
xsendmail       67     0                       y|Y|n|N) ;;
xsendmail       68     0                       *)  echo " ";
xsendmail       69     0                           echo "      You must enter \"y\" for yes and \"n\  for no.";
xsendmail       70     0                           y=?;;
xsendmail       71     0               esac
xsendmail       72     0       done
xsendmail       73     0
xsendmail       74     0       case $y in
xsendmail       75     0               y|Y)  echo " ";
xsendmail       76     0                     echo -n "What is the name of the file you wish to send?  ";
xsendmail       77     0                    read y;
xsendmail       78     0                    if /bin/test -f "$y
xsendmail       79     0                    then
xsendmail       80     0                       echo "
xsendmail       81     0                       "
xsendmail       82     0                       echo "!!! mail $x < $y"
xsendmail       83     0                       echo "
xsendmail       84     0                       "
xsendmail       85     0                            .
xsendmail       86     0                       mail $x < $y
xsendmail       87     0                    else
xsendmail       88     0                       echo
xsendmail       89     0                       echo "File $y does not exist!"
xsendmail       90     0                       sleep 2
xsendmail       91     0                       exit 0
xsendmail       92     0                     fi;;
xsendmail       93     0               n|N)  echo " ";
xsendmail       94     0                     echo "      After you see the UNIX command enter your message one line at ";
xsendmail       95     0                     echo "a time.  When you have finished entering your message press the";
xsendmail       96     0                     echo "CONTROI and \"D\" keys simultaneously.";
xsendmail       97     0                     echo "
xsendmail       98     0                           ";
xsendmail       99     0                     echo "!!! mail $x";
xsendmail      100     0                     echo "
```

```
file          line level  source

xsendmail      101     0
xsendmail      102     0                              "
xsendmail      103     0                          mail $x;;
xsendmail      104     0       esac
xsendmail      105     0
xsendmail      106     0
xsendmail      107     0       echo '
xsendmail      108     0
xserdmail      109     0       echo -n "Press return tc continue
xsendmail      110     0
xsendmail      111     0       read z
xsendmail      112     0
xsendmail      113     0       exit 0
```

```
file          line level  source

xtempexit        1     0
xtempexit        2     0      # xtempexit
xtempexit        3     0      #
xtempexit        4     0      #       Purpose - To provide a means of temporarily exiting the menu
xtempexit        5     0      #                 shell.
xtempexit        6     0      #
xtempexit        7     0      #       Method -  The program consists of a loop which echos a prompt
xtempexit        8     0      #                 to the user and waits for input from the keyboard.
xtempexit        9     0      #                 The user's entry is executed.  The loop continues
xtempexit       10     0      #                 until the user enters a null response.
xtempexit       11     0      #
xtempexit       12     0      #       Error checking - The program does no error checking.  If the
xtempexit       13     0      #                 user enters an incorrect UNIX command, he is presented
xtempexit       14     0      #                 the standard UNIX error messages.
xtempexit       15     0      #
xtempexit       16     0      x=?
xtempexit       17     0      echo '
xtempexit       18     0
xtempexit       19     0                          TEMPORARILY EXIT MENU SHELL
xtempexit       20     0
xtempexit       21     0
xtempexit       22     0           You are able to return to the menu shell any time by pressing the
xtempexit       23     0      the return key.  Proceed to enter the UNIX commands you wish to execute.
xtempexit       24     0
xtempexit       25     0           '
xtempexit       26     0
xtempexit       27     0      while /bin/test "$x"
xtempexit       28     0
xtempexit       29     0      do
xtempexit       30     0              echo '
xtempexit       31     0
xtempexit       32     0              echo -n "$ '
xtempexit       33     0              read x
xtempexit       34     0
xtempexit       35     0              eval "$x'
xtempexit       36     0      done
xtempexit       37     0
xtempexit       38     0      exit 0
```

```
file          line level  source

xterminal        1     0
xterminal        2     0      # xterminal
xterminal        3     0      #
xterminal        4     0      #       Purpose - To introduce the user to special keys on his terminal.
xterminal        5     0      #
xterminal        6     0      #       Method -  Xterminal displays files which describe the special
xterminal        7     0      #                 keys available on the terminals used at RIT.  The
xterminal        8     0      #                 files are displayed using the "more" command.
xterminal        9     0      #                 This shell script uses the "tset" command to identify
xterminal       10     0      #                 the current terminal.  A case statement is used
xterminal       11     0      #                 to display the appropriate file.  The case statement
xterminal       12     0      #                 can be expanded to accommodate additional terminals.
xterminal       13     0      #
xterminal       14     0      echo "
xterminal       15     0
xterminal       16     0
xterminal       17     0                                  INTROLUCTION TO TERMINAL
xterminal       18     0
xterminal       19     0
xterminal       20     0           There are keys on your terminal that have special meanings.  If
xterminal       21     0      you use these keys your time spent working at your terminal will become
xterminal       22     0      more productive.
xterminal       23     0
xterminal       24     0           This menu option recognizes the keys for the most common terminals
xterminal       25     0      used in the Computer Science Department at Rochester Institute of Tech-
xterminal       26     0      nology.
xterminal       27     0
xterminal       28     0
xterminal       29     0      sleep 5
xterminal       30     0
xterminal       31     0      x=`tset -`
xterminal       32     0      case $x in
xterminal       33     0             gigigigi)
xterminal       34     0                     cat term_gigi | more -d;;
xterminal       35     0             *)
xterminal       36     0                     cat term_misc | more -d;;
xterminal       37     0      esac
xterminal       38     0
xterminal       39     0      echo -n "Press return to continue "
xterminal       40     0      read x
xterminal       41     0
xterminal       42     0      exit 0
```

```
file          line level  source

xwho             1     0
xwho             2     0       #xwho
xwho             3     0       #
xwho             4     0       #       Purpose - To provide a means for displaying the current
xwho             5     0       #                 system users.
xwho             6     0       #
xwho             7     0       #       Method -  A brief explanation of the "who" command is
xwho             8     0       #                 given.  The shell then issues the "who" command.
xwho             9     0       #
xwho            10     0       echo
xwho            11     0
xwho            12     0                               WHO IS WORKING ON THE SYSTEM
xwho            13     0
xwho            14     0
xwho            15     0            The following command lists the login name, terminal name and
xwho            16     0       login time for each user currently on the system:
xwho            17     0
xwho            18     0
xwho            19     0       !!! who
xwho            20     0
xwho            21     0
xwho            22     0
xwho            23     0       sleep 5
xwho            24     0       who
xwho            25     0
xwho            26     0       echo '
xwho            27     0
xwho            28     0       echo -n "Press return to continue
xwho            29     0       read z
xwho            30     0
xwho            31     0       exit 0
```

```
file          line level  source

xwrite           1     0
xwrite           2     0      #xwrite
xwrite           3     0      #
xwrite           4     0      #       Purpose - To provide a means to write to another user.
xwrite           5     0      #
xwrite           6     0      #       Method -  If the level of prompt is verbose, an explanation
xwrite           7     0      #                 of the "write" command is given.  The explanation
xwrite           8     0      #                 includes a description of the protocol to be
xwrite           9     0      #                 followed.  The user must enter the login name of the
xwrite          10     0      #                 user to whom he wishes to write.  If the user enters
xwrite          11     0      #                 a return, the program will exit.
xwrite          12     0      #
xwrite          13     0      question=?
xwrite          14     0      x=?
xwrite          15     0
xwrite          16     0      echo '
xwrite          17     0
xwrite          18     0                                  WRITE TO ANOTHER USER
xwrite          19     0
xwrite          20     0            '
xwrite          21     0
xwrite          22     0      if /bin/test $1 = 1
xwrite          23     0      then
xwrite          24     0              echo '     The \"write\' command is used to communicate with another user"'
xwrite          25     0              echo 'currently logged on to the system.  The messages sent with \"write\""'
xwrite          26     0              echo "will be displayed at the receiver's terminal immediately.  He may wish"'
xwrite          27     0              echo "to respond to you by writing back, thus establishing two way communi-"'
xwrite          28     0              echo "cation.  For this reason the following protocol has been established:"'
xwrite          29     0              echo '
xwrite          30     0              echo '   1)  When you see the UNIX command enter your message followed by'
xwrite          31     0              echo '       a \"-o\" which indicates you have temporarily finished'
xwrite          32     0              echo '       writing."'
xwrite          33     0              echo '   2)  Wait for the receiver to answer back.  You will know he is'
xwrite          34     0              echo '       finished when you see his \"-o\".  If you do not expect"'
xwrite          35     0              echo '       an answer, skip to step 5."'
xwrite          36     0              echo '   3)  Continue writing back and forth in the above manner.'
xwrite          37     0              echo '   4)  End your final message with \"-oo\" to indicate your communi-'
xwrite          38     0              echo '       cation has completed."'
xwrite          39     0              echo '   5)  Press the CONTROL and \"D\" keys simultaneously to discontinue'
xwrite          40     0              echo '       the writing process."'
xwrite          41     0              echo " '
xwrite          42     0      fi
xwrite          43     0
xwrite          44     0      while /bin/test "$x" = "$question"
xwrite          45     0      do
xwrite          46     0              echo ' '
xwrite          47     0              case $1 in
xwrite          48     0                      1)  echo -n "*What is the login name of the person to whom you wish to write?  ";;
xwrite          49     0                      2)  echo -n "*To whom?
xwrite          50     0              esac
```

```
file            line level  source

xwrite            51     0
xwrite            52     0                read x
xwrite            53     0
xwrite            54     0                if /bin/test "$x"
xwrite            55     0                then
xwrite            56     0                        if /bin/test "$x" = '$question'
xwrite            57     0                        then
xwrite            58     0                                echo ' '
xwrite            59     0                                echo "
xwrite            60     0                                echo "     You must enter the login name of someone currently using the"
xwrite            61     0                                echo "system."
xwrite            62     0                        fi
xwrite            63     0                else
xwrite            64     0                        exit 0
xwrite            65     0                fi
xwrite            66     0        done
xwrite            67     0
xwrite            68     0        echo '
xwrite            69     0        Remember - \"CONTROL D\" discontinues the write process!
xwrite            70     0
xwrite            71     0        !!! write $x
xwrite            72     0
xwrite            73     0        write $x
xwrite            74     0
xwrite            75     0        echo '
xwrite            76     0
xwrite            77     0        echo -n "Press return to continue "
xwrite            78     0        read z
xwrite            79     0
xwrite            80     0        exit 0
```

```
file          line level  source

term_misc        1     0
term_misc        2     0
term_misc        3     0
term_misc        4     0
term_misc        5     0
term_misc        6     0
term_misc        7     0
term_misc        8     0          To temporarily stop the screen press the "CONTROL" key and
term_misc        9     0     the "S" key simultaneously. When you are ready to restart the
term_misc       10     0     screen movement, press the "CONTROL" and "Q" keys simultaneously.
term_misc       11     0
term_misc       12     0
term_misc       13     0
term_misc       14     0
term_misc       15     0
term_misc       16     0
term_misc       17     0
term_misc       18     0
term_misc       19     0
term_misc       20     0
term_misc       21     0
term_misc       22     0
term_misc       23     0
term_misc       24     0
term_misc       25     0
term_misc       26     0
term_misc       27     0
term_misc       28     0          The "TAB" key will move the cursor 8 positions to the right
term_misc       29     0     each time it is pressed. If your terminal does not have the
term_misc       30     0     "TAB" key, you can achieve the same result by pressing the "CON-
term_misc       31     0     TROL" and "I" keys simultaneously.
term_misc       32     0
term_misc       33     0
term_misc       34     0
term_misc       35     0
term_misc       36     0
term_misc       37     0
term_misc       38     0
term_misc       39     0
term_misc       40     0
term_misc       41     0
term_misc       42     0
term_misc       43     0
term_misc       44     0
term_misc       45     0
term_misc       46     0
term_misc       47     0
term_misc       48     0          If you make a mistake when entering a line there are two
term_misc       49     0     options for making corrections. These options only apply to
term_misc       50     0     errors in the current line.
```

file line level source

term_misc 51 0
term_misc 52 0 If your terminal has a backspace key, simply press that
term_misc 53 0 key until the cursor has backed up to the error position.
term_misc 54 0 If you do not have a backspace key, press the "CONTROL"
term_misc 55 0 and "H" keys simultaneously to back up the cursor. Once
term_misc 56 0 the cursor is in the correct position, type in the
term_misc 57 0 corrected line.
term_misc 58 0
term_misc 59 0
term_misc 60 0 Sometimes a line has many errors and you feel it would be
term_misc 61 0 easier to retype the entire line. To have the computer
term_misc 62 0 ignore the current line press the "CONTROL" and "U" keys
term_misc 63 0 simultaneously.
term_misc 64 0
term_misc 65 0
term_misc 66 0
term_misc 67 0
term_misc 68 0
term_misc 69 0
term_misc 70 0
term_misc 71 0
term_misc 72 0 There will be times when you want to stop the current program.
term_misc 73 0 The "BREAK" key or the "DELETE" key can be used to stop the
term_misc 74 0 current program. If you do not see the prompt (the signal that
term_misc 75 0 the computer is ready for your next command) you may have to
term_misc 76 0 press the "RETURN" key to have the prompt appear.
term_misc 77 0
term_misc 78 0
term_misc 79 0
term_misc 80 0
term_misc 81 0
term_misc 82 0
term_misc 83 0
term_misc 84 0
term_misc 85 0
term_misc 86 0
term_misc 87 0
term_misc 88 0
term_misc 89 0
term_misc 90 0
term_misc 91 0
term_misc 92 0
term_misc 93 0
term_misc 94 0
term_misc 95 0
term_misc 96 0
term_misc 97 0
term_misc 98 0
term_misc 99 0
term_misc 100 0

file	line	level	source
term_gigi	1	0	
term_gigi	2	0	
term_gigi	3	0	
term_gigi	4	0	
term_gigi	5	0	
term_gigi	6	0	
term_gigi	7	0	
term_gigi	8	0	To temporarily stop the screen press the "NO SCROLL" key.
term_gigi	9	0	When you are ready to restart the screen movement press "NO
term_gigi	10	0	SCROLL" again.
term_gigi	11	0	
term_gigi	12	0	
term_gigi	13	0	
term_gigi	14	0	
term_gigi	15	0	
term_gigi	16	0	
term_gigi	17	0	
term_gigi	18	0	
term_gigi	19	0	
term_gigi	20	0	
term_gigi	21	0	
term_gigi	22	0	
term_gigi	23	0	
term_gigi	24	0	
term_gigi	25	0	
term_gigi	26	0	
term_gigi	27	0	
term_gigi	28	0	The "TAB" key will move the cursor 8 positions to the right
term_gigi	29	0	each time it is pressed. If your terminal does not have the
term_gigi	30	0	"TAB" key, you can achieve the same result by pressing the "CON-
term_gigi	31	0	TROL" and "I" keys simultaneously.
term_gigi	32	0	
term_gigi	33	0	
term_gigi	34	0	
term_gigi	35	0	
term_gigi	36	0	
term_gigi	37	0	
term_gigi	38	0	
term_gigi	39	0	
term_gigi	40	0	
term_gigi	41	0	
term_gigi	42	0	
term_gigi	43	0	
term_gigi	44	0	
term_gigi	45	0	
term_gigi	46	0	
term_gigi	47	0	
term_gigi	48	0	If you make a mistake when entering a line there are two
term_gigi	49	0	options for making corrections. These options only apply to
term_gigi	50	0	errors in the current line.

file	line	level	source
term_gigi	51	0	
term_gigi	52	0	If your terminal has a backspace key, simply press that
term_gigi	53	0	key until the cursor has backed up to the error position.
term_gigi	54	0	If you do not have a backspace key, press the "CONTROL"
term_gigi	55	0	and "H" keys simultaneously to back up the cursor. Once
term_gigi	56	0	the cursor is in the correct position, type in the
term_gigi	57	0	corrected line.
term_gigi	58	0	
term_gigi	59	0	
term_gigi	60	0	Sometimes a line has many errors and you feel it would be
term_gigi	61	0	easier to retype the entire line. To have the computer
term_gigi	62	0	ignore the current line press the "CONTROL" and "U" keys
term_gigi	63	0	simultaneously.
term_gigi	64	0	
term_gigi	65	0	
term_gigi	66	0	
term_gigi	67	0	
term_gigi	68	0	
term_gigi	69	0	
term_gigi	70	0	
term_gigi	71	0	There will be times when you want to stop the current program.
term_gigi	72	0	The "BREAK" key or the "DELETE" key can be used to stop the
term_gigi	73	0	current program. If you do not see the prompt (the signal that
term_gigi	74	0	the computer is ready for your next command) you may have to
term_gigi	75	0	press the "RETURN" key to have the prompt appear.
term_gigi	76	0	
term_gigi	77	0	
term_gigi	78	0	
term_gigi	79	0	
term_gigi	80	0	
term_gigi	81	0	
term_gigi	82	0	
term_gigi	83	0	
term_gigi	84	0	
term_gigi	85	0	
term_gigi	86	0	
term_gigi	87	0	
term_gigi	88	0	
term_gigi	89	0	
term_gigi	90	0	
term_gigi	91	0	
term_gigi	92	0	
term_gigi	93	0	
term_gigi	94	0	
term_gigi	95	0	
term_gigi	96	0	
term_gigi	97	0	
term_gigi	98	0	
term_gigi	99	0	
term_gigi	100	0	

file line level source

introduction 1 0
introduction 2 0
introduction 3 0
introduction 4 0 Welcome to the Menu shell. It has been designed to make the
introduction 5 0 UNIX operating system friendlier to you the user. UNIX is a
introduction 6 0 powerful operating system which you will appreciate as you become
introduction 7 0 familiar with it. UNIX does have a limitation in regard to the
introduction 8 0 new user. UNIX is a very terse operating system and offers lit-
introduction 9 0 tle help to its users. Once you are familiar with UNIX commands
introduction 10 0 the limitation of terseness becomes a desirable feature.
introduction 11 0
introduction 12 0
introduction 13 0
introduction 14 0
introduction 15 0
introduction 16 0
introduction 17 0
introduction 18 0
introduction 19 0
introduction 20 0
introduction 21 0
introduction 22 0
introduction 23 0
introduction 24 0
introduction 25 0
introduction 26 0 The purpose of the Menu shell is threefold:
introduction 27 0
introduction 28 0 1. Allows you to function in the UNIX environment
introduction 29 0 successfully without knowing any UNIX commands.
introduction 30 0 2. Serves as a tool for learning UNIX commands.
introduction 31 0 3. Serves as a review to users who have not used
introduction 32 0 UNIX for an extended period of time.
introduction 33 0
introduction 34 0
introduction 35 0
introduction 36 0
introduction 37 0
introduction 38 0
introduction 39 0
introduction 40 0
introduction 41 0
introduction 42 0
introduction 43 0
introduction 44 0
introduction 45 0
introduction 46 0
introduction 47 0
introduction 48 0
introduction 49 0 The Menu shell includes an option which introduces you to
introduction 50 0 your terminal. It would be beneficial to choose this option if

```
file          line level  source

introduction    51     0      you are unfamiliar with the terminal you are using.  This  option
introduction    52     0      will identify special keys and their meanings.
introduction    53     0
introduction    54     0
introduction    55     0
introduction    56     0
introduction    57     0
introduction    58     0
introduction    59     0
introduction    60     0
introduction    61     0
introduction    62     0
introduction    63     0
introduction    64     0
introduction    65     0
introduction    66     0
introduction    67     0
introduction    68     0
introduction    69     0
introduction    70     0
introduction    71     0           Whenever you see three exclamation marks (!!!) on the screen
introduction    72     0      it  means  that  the command which follows is the UNIX command to
introduction    73     0      perform the particular task you asked for.  For example:
introduction    74     0
introduction    75     0          !!! date
introduction    76     0
introduction    77     0      indicates to you that "date" is a valid  UNIX  command.   If  you
introduction    78     0      entered  "date" at your terminal it would respond with the current
introduction    79     0      date and time.
introduction    80     0
introduction    81     0
introduction    82     0
introduction    83     0
introduction    84     0
introduction    85     0
introduction    86     0
introduction    87     0
introduction    88     0
introduction    89     0
introduction    90     0
introduction    91     0
introduction    92     0
introduction    93     0           The Menu shell provides for  two  levels  of  prompting  you
introduction    94     0      regarding  the  tasks  you wish to perform.  The default level of
introduction    95     0      prompt is verbose and the most beneficial to use  in  the  begin-
introduction    96     0      ning.   After  you have spent some time in the Menu shell and are
introduction    97     0      familiar with the various tasks  it  offers,  you  may  find  the
introduction    98     0      second  level  of  prompting  to be best.  You are able to switch
introduction    99     0      back and forth from the verbose and terse prompts by choosing the
introduction   100     0      appropriate menu option.
```

```
file          line level  source

introduction   101    0
introduction   102    0
introduction   103    0
introduction   104    0
introduction   105    0
introduction   106    0
introduction   107    0
introduction   108    0
introduction   109    0
introduction   110    0
introduction   111    0
introduction   112    0
introduction   113    0
introduction   114    0          You will be asked questions which are needed to perform your
introduction   115    0      desired tasks.  Most of the questions are preceded by an "*".
introduction   116    0      Any time an "*" appears it is a signal to you that you can obtain
introduction   117    0      more information by responding with a "?" Once you have seen the
introduction   118    0      additional explanation you will be asked the same question again.
introduction   119    0      It would be especially beneficial for you to respond with a "?"
introduction   120    0      to the "*" question the first time you have a task performed.
introduction   121    0
introduction   122    0
introduction   123    0
introduction   124    0
introduction   125    0
introduction   126    0
introduction   127    0
introduction   128    0
introduction   129    0
introduction   130    0
introduction   131    0
introduction   132    0
introduction   133    0
introduction   134    0
introduction   135    0
introduction   136    0
introduction   137    0          If you select a menu option and realize that you really do
introduction   138    0      not want that task performed, the best way to exit that task is
introduction   139    0      to simply enter a "return" to the first question you are asked.
introduction   140    0
introduction   141    0
introduction   142    0
introduction   143    0
introduction   144    0
introduction   145    0
introduction   146    0
introduction   147    0
introduction   148    0
introduction   149    0
introduction   150    0
```

file	line	level	source
introduction	151	0	
introduction	152	0	
introduction	153	0	
introduction	154	0	
introduction	155	0	
introduction	156	0	
introduction	157	0	As you become familiar with UNIX you will find the option to
introduction	158	0	temporarily exit the Menu shell extremely useful. This option
introduction	159	0	allows to use any of the UNIX commands you know on your own. You
introduction	160	0	will be able to quickly perform tasks without the menu structure.
introduction	161	0	When you wish to return to the menus again simply press the
introduction	162	0	return key.
introduction	163	0	
introduction	164	0	GOOD LUCK!!
introduction	165	0	
introduction	166	0	
introduction	167	0	
introduction	168	0	
introduction	169	0	
introduction	170	0	
introduction	171	0	
introduction	172	0	
introduction	173	0	
introduction	174	0	
introduction	175	0	
introduction	176	0	
introduction	177	0	
introduction	178	0	
introduction	179	0	
introduction	180	0	