Teleconferencing and the MARRS Computer Conferencing System

by

THOMAS JOSEPH STACHOWICZ

B. S. E. E., The University of Toledo, 1980

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

Approved by:

[illegible]

Major Professor

CONTENTS

LIST OF FIGURES

Acknowledgements

I would like to dedicate this work to my wife, Maggie, and daughter, Laura. Their love and understanding allowed all this to be possible. I would especially like to thank my advisor, Dr. Rich McBride, for the many hours of time spent in suggesting ideas and guiding this work.

1. Introduction

This paper provides a detailed overview of teleconferencing and discusses the design of an asynchronous computer teleconferencing system, MARRS (MAster's Report Reviewing System). This system is applicable to automating the process of collecting, reviewing and presenting of Master's projects.

An asynchronous computer teleconferencing system was chosen over other forms of teleconferencing because 1) a computer system was readily available, 2) a computer system is easily applicable to such a teleconferencing system, and 3) computer teleconferencing allows for asynchronous, "store and forward" communication, where all participants need not be present at the same time for the same conference. The use of an asynchronous computer teleconferencing system allows meetings, discussions, reviews, etc., without gathering people at one place at one time.

The main focus of this paper is on the various roles of the participants in a MARRS conference. Prior to this discussion, teleconferencing and its various types are discussed so that the reader will be familiar with a variety of teleconferencing forms and applications.

This paper is organized in four chapters. The introduction highlights the main issues in this paper. Chapter 2 is a review of literature containing a detailed overview of teleconferencing in its many forms. Audio, audiographics, video, and computer teleconferencing are discussed. Examples of actual uses of each type are given.

MARRS, an asynchronous computer teleconferencing system is described in chapter 3. The various roles of each participant in a conference are discussed.

The final chapter discusses conclusions, enhancements, and the future work related to the MARRS teleconferencing system and others.

2. Literature Review

2.1 Overview

Teleconferencing can be defined as any form of electronic interactive communication. Teleconferencing can be found in several forms, all providing some form of an "electronic meeting".

In this paper, four major forms of teleconferencing will be discussed. They are:

- Audio Teleconferencing,
- Audiographics Teleconferencing,
- Video Teleconferencing and
- Computer Teleconferencing.

Audio teleconferencing is voice-only communication. Audiographics includes audio teleconferencing with the addition of graphic or written illustrations. Video teleconferencing includes the features just mentioned in audiographics, and also images of people. Finally, computer teleconferencing provides a quite different approach by using a computer to link the users.

Although these four types of teleconferencing are different in nature, complexity, and costs, they do have several common features. First of all, they link people together at different locations using some type of telecommunications technology. Secondly, they provide two-way interactive communication. Finally, they are dynamic, involving active participation of people.

Beyond these similarities lie many differences which allow one form of teleconferencing to be chosen over another for certain applications. In the following sections, the types of teleconferencing will be discussed in detail. The strengths and weaknesses of each type will be included along with examples of practical applications.

2.2 Audio Teleconferencing

Audio teleconferencing is limited to voice-only communication. With this type of teleconferencing, people at different locations can be linked together via ordinary telephone lines. Covered under this type of "electronic meeting" is a simple conference call as well as more complex systems bridging together multiple locations.

2.2.1 Forms of Audio Teleconferencing Several forms of audio teleconferencing exist, exhibiting a number of advantages and disadvantages.

2.2.1.1 User-initiated The user-initiated conference call is used more frequently than any other form of audio teleconferencing because of its simplicity and convenience. With a telephone in hand, a user can have several locations linked together using the public telephone network. With this form of audio teleconferencing, however, the user has little or no control over audio quality.

2.2.1.2 Dial-out A second form of audio teleconferencing uses dial-up or dial-out bridges. A central bridge is used and the calls are usually dialed by an operator and linked together. This form usually allows up to 10 sites to be linked together using the public telephone network [17]. Some dial-out bridges offer audio control adjustments to allow for better voice quality. Dial-out teleconferencing becomes an unattractive alternative when large time delays are encountered in setting up overcrowded dial-up teleconferences.

2.2.1.3 Meet-me Dial-in or meet-me teleconferencing, the third form of audio teleconferencing, is similar to dial-out teleconferencing. The major difference is that users call a dial-in bridge at a specified time rather than waiting to be called by a dial-out operator. This form has become very popular due to its flexibility, audio control quality (manual or automatic), and number of possible conference sites. An individual bridge of this type may link 10 to 25 locations or over 200 sites when used in tandem [16]. Meet-me teleconferencing may use, in addition to the public telephone network, WATS (Wide Area Telephone Service) in order to help control communication costs. A disadvantage, not found in dial-out teleconferencing, is that a user who is not prompt in calling at the predetermined time can hold up a conference. Higher communication costs result in this case and other users are likely to be upset.

Because of its popularity, several companies now offer this type of bridging service. A company that uses audio conferencing, but not extensively enough to justify the purchase of its own equipment, may choose to rent time from a commercial company offering such a service. The service may also include other options such as equipment rental or tape recording. An organization, on the other hand, may choose to purchase its own equipment and sell time to others to justify its purchase price.

2.2.1.4 Dedicated A final form of audio teleconferencing, dedicated networks, involves leasing transmission circuits to form a network among specific locations. This form is usually of better voice quality because of the control it offers over audio transmission. It may be used for data transmission. Depending on the frequency and length of audio teleconferencing use, dedicated networks may be most cost-effective. Public telephone lines may be tied into the network to allow more flexibility for sites not located on the network.

2.2.2 Strengths and Weaknesses of Audio Teleconferencing Many advantages of audio teleconferencing exist. They may not apply, however, in all cases. The company who has reduced travel expenses by using audio teleconferencing contrasts to the organization who has seen costs rise due to increased interaction (due to teleconferencing) among employees and clients. In general, however, it is realistic to expect travel expenses and wasted time (flying, driving, waiting, etc.) to decrease in most cases.

The importance of good audio in such a system cannot be overemphasized. When audio is all there is (not to imply that it is simple), it is extremely important to maintain good quality to avoid distractions in a meeting. As Johansen and Bullen state, "Slightly fuzzy pictures or participants or less crisp letters in viewgraphs will not ruin a meeting; but the inability to hear clearly what is being said will" [13].

Audio systems can be used effectively in place of face-to-face meetings. There are times, however, when face-to-face contact is more advantageous. This may be when 1) trying to make a good impression (one may not want to appear "cheap"), 2) dealing with new people or occasionally renewing old acquaintances, 3) nonverbal cues are important in the meeting or 4) trying to put the pressure on someone. It is important to recognize when audio teleconferencing should not be used so that the merits or it are not downgraded.

With audio teleconferencing, it may be difficult to identify who is speaking and seemingly redundant to state one's name before speaking each time. In the following section on audiographics, it is noted how this problem can be helped.

With an audio system, barriers between an employee and management or between different levels or management may tend to diminish. When this happens, better, more effective communication in the organization may result. Finally, audio teleconferencing usually results in shorter meetings which can be set up very quickly when necessary.

2.2.3 <u>Example Uses of Audio Teleconferencing</u> The Bank of America divided its headquarters between San Francisco and Los Angeles in the late 1950's. The hassles of commuting between cities resulted in the company considering a teleconferencing system. A video system was deemed too expensive and in 1968 a high-quality audio teleconferencing system was installed.

At first the needed high audio quality eluded the system. After much added design work and experimentation, the system performed excellently and was well excepted and heavily used.

NASA officials spent much of their time traveling during the development of Apollo. To reduce travel and related expenses, NASA had an audio teleconferencing system installed in the late 1960's. The network connected 31 NASA and contractor sites across the United States [18].

Poor upkeep and maintenance on part or the system kept it from being an excellent teleconferencing system. However, it was and still is being used for conferencing today.

2.3 Audiographics Teleconferencing

Audiographics is another form of teleconferencing. Audiographics go a step further than audio teleconferencing by supplementing voice transmission with written or graphic information. All audiographics systems transmit visual information over narrowband channels.

2.3.1 Forms of Audiographics Teleconferencing

Several forms of audiographics teleconferencing systems exist. The differences in the capability of each is discussed below.

2.3.1.1 <u>Facsimile</u> The facsimile machine is the most common of all audiographics devices. With this device, a document's (a full-sized page) image is converted into electronic signals. These signals are transmitted to another facsimile machine at a remote end, recreating the document. Digital machines can transmit a page in less than a minute while most analog machines take several minutes.

In some cases the transmission delay of a document can be an annoyance. When possible transmission should be done before a meeting.

2.3.1.2 <u>Telewriters</u> Telewriters include electronic blackboards or tablets, light pens, and other similar devices. They are used in order to transmit hand-drawn writing, figures, equations, etc. in real time. At one end, an image is produced electronically using a light pen and/or a special surface. The image is then transmitted to the remote location where it is displayed on a monitor (or recorded). This form of audiographics teleconferencing is best suited when real-time graphic interaction is needed.

2.3.1.3 <u>Computer Systems</u> Using a computer system as an adjunct to an audio teleconferencing system can be implemented with common computer technology. Graphics, tables, database retrievals, etc. can be displayed on a computer terminal as an added feature of a video conference. Ideally this is done in real-time, however the computer information could be printed or copied prior to the conference.

2.3.1.4 Others A fairly new device applicable to audiographics is a random access microfiche or slide projector [17]. With this device signals are generated at a remote site and transmitted over a telephone line. When received, the signals are interpreted and the selected slide or fiche is displayed.

In order to try to eliminate the ambiguity of who is remotely speaking at an audiographics teleconference, several approaches have been devised. In the British approach, called the Remote Meeting Table (RTM), a separate speaker (with an attached light) is used to generate the sound of each remote person. By using directional microphones, a participant at the remote end can be identified by the speaker which produces the persons voice (the light is lit on the speaker as a visual confirmation of the speaker's operation). Names can be placed on the speakers, if needed [18]. The French have used an LED display alongside a list of participants' names. When speaking, the microphone is able to determine who is speaking and subsequently lights the correct LED on the name board.

Bell Laboratories has used a system to mock the raising of one's hand at a conference. Each participant is given a number which is keyed into a number pad (a tone generator) or push-button phone when wanting to enter the discussion. This identification can be used for queuing people for discussion as well as for identification to others in the conference [15].

The different forms of audiographics systems can be combined in order to form a system to suit the users' needs.

2.3.2 Strengths and Weaknesses of Audiographics Teleconferencing

Several advantages exist for this type of teleconferencing. The added options or graphic capabilities give the system an advantage over straight audio teleconferencing when considering information transmission capabilities. In most cases, a more complete conference presentation is possible with audiographics teleconferencing.

Although it is usually economical, audiographics is more expensive than audio teleconferencing. For that reason, audiographics should be used over audio when the added benefit justifies the added cost.

2.3.3 Example Use of Audiographics Teleconferencing Honeywell Inc.'s audiographics teleconferencing system uses a 3M EMT 9140 facsimile unit to transmit documents and other conference material -- and is saving $150,000 a month in travel expenses [23]. In addition to the financial savings, employee productivity has increased.

Due to the high transportation costs for employees (and predictions for increases to continue), Honeywell instituted a formal program for teleconferencing. Six new audiographics conference rooms were designed and built to include the AT&T "Electronic Blackboard." Transmission of the audio, facsimiles, and "still" video is done with Honeywell's private voice network.

Honeywell employees are now missing out on the "wear and tear" of traveling, while conducting meetings that get to the point quicker.

They have also been using audiographics teleconferencing to supplement certain infrequent face-to-face meetings.

2.4 Video Teleconferencing

Video is a third form of teleconferencing. Video systems include voices, graphics and pictures of people. Video teleconferencing allows the speaker to be seen and heard at the same time.

2.4.1 Forms of Video Teleconferencing Several forms of video teleconferencing exist, differing widely in cost and quality. Video teleconferencing to many means full-color television. But this full-motion video is only one form, the other two being freeze-frame and compressed video. These will be discussed in the following sections.

2.4.1.1 Freeze-frame In freeze-frame video, a "slice" of a moving picture is generated. This "frozen" picture can then be sent over a public telephone line to any desired location. This form of video is used chiefly to display three-dimensional or graphic-type images much more economically than with the other two forms. Video teleconferencing is sometimes classified as an audiographic technology because of its similarity in sending/displaying objects or graphics.

Freeze-frame video can be used for black & white or color monitors and takes sseconds to transmit images (because of the narrow bandwidth of a regular telephone line). As previously mentioned, it can be done at a fraction of the cost of compressed or full-motion video.

2.4.1.2 Compressed Compressed video uses one of the newest video technologies called codec. Because much of the information in a full-motion video picture is redundant, a codec uses a compression technique in order to eliminate much of the redundant information. Once eliminated, the resulting compressed video can be transmitted over a much narrower transmission channel at a much lower cost. An original video signal of 100 million bits is reduced to a signal of 1.5 or 3 million bits [17]. The "new" picture looks very much like the original with the exception of some occasional blurring or jerking. Future codecs hope to eliminate this mild drawback.

Compressed video, usually used for color pictures, uses a T1 data circuit to transmit the image instantly. Although a codec can greatly reduce transmission costs, the cost of one is still high (about $100,000). Future advances in video codecs will improve image quality and lower their price.

2.4.1.3 Full-motion Full-motion video systems use wideband channels to send video, data, and voice signals. They use 6 megaHertz and are comparable to broadcast television in quality. With such a large channel capacity, it can transmit the continuous motion in real time. Full motion video is expensive. Because of this it may fade away to a cheaper "compressed video" with little, if any, observable differences.

2.4.2 Strengths and Weaknesses of Video Teleconferencing Video teleconferencing offers the closest thing to face-to-face meetings. However, in some cases it cannot be a substitute for a face-to-face encounter. Many times video teleconferencing is perceived to be more effective and satisfying than other forms of teleconferencing, especially if the participants do not know each other.

Other than in face-to-face meetings, eye contact is possible only with video. In addition, video meetings often tend to follow a more orderly fasnion than face-to-face meetings [16].

Video teleconferencing is expensive if full-motion is used. It is not as expensive for freeze-frame video (if it can be used for the required task). For companies without their own video teleconferencing system but requiring occasional video teleconferencing, the trip to the video teleconferencing studio can prove to be somewhat unpopular.

2.4.3 Example Uses of Video Teleconferencing IBM has been using its own in-house "still" video teleconferencing system since 1975. The system uses audio conferencing to go with the images displayed on the television monitors. It is mostly used by the technical staff and has been found to be very useful.

Full-motion video conferencing has been used, since 1981, by Aetna Life and Casualty's data processing staff. The system links their offices in Hartford and nearby Windsor, Connecticut. It has been so successful that Aetna is now expanding its video capability to other parts of the country.

For those organizations without their own video teleconferencing facilities, AT&T rents fully equipped rooms in 14 cities. Voice sensitive cameras operate so as to record the person who is speaking. The service can cost up 48 per hour for the room and network access. Advances in the satellite and fiber optic field could bring the cost down to under $100, according to a technology forecasting firm, Future Systems, Inc. [2].

Top American and Japanese trade officials have recently "met" using the video teleconferencing facilities of BizNet, the U.S. Chamber of Commerce's television system. Full-motion video between Tokyo and Washington for this unprecedented dialogue was carried over five satellites [18].

Future teleconferencing systems may use holography [2]. Holography uses laser technology to project life-like images into the air. Using this technology, video teleconferencing would approach the limits of face-to-face meetings.

2.5 Computer Teleconferencing

Computer teleconferencing, a fourth type of teleconferencing, differs from the rest in that voice communication is not used. This most recent addition to the teleconferencing field generally uses a central computer to store the complex dialogue of a group of people. This type of teleconferencing uses public telephone lines and is therefore very flexible. One's geographic location is not a limitation.

2.5.1 Forms of Computer Teleconferencing

2.5.1.1 Synchronous A synchronous teleconference is one that is conducted in real time -- that is, everyone is present at their own location at the same time. Synchronous computer teleconferencing uses a keyboard for information entry and can tie together many computer users. The dialogue of a user can be seen by the intended recipients as it is entered at the keyboard. However, most computer conferencing refers to the "asynchronous" type.

2.5.1.2 Asynchronous An asynchronous teleconference is one that is conducted without the restriction of everyone "meeting" at the same time. The inherent data storing nature of a computer makes it ideal for such a "store and forward" system. Therefore asynchronous computer teleconferencing is not restricted by location or time.

2.5.2 Strengths and Weaknesses of Computer Teleconferencing Computer teleconferencing's biggest advantage is that it overcomes geographic and time constraints. This type of teleconferencing is self-documenting so meeting notes or minutes do not have to be documented or reinterpreted.

Computer teleconferencing offers many features. These features are not available on all systems but can be added by changes in software. Some of the feature include 1) online meetings with a conference management system, 2) personal work areas that can be used like a notepad, 3) online bulletin boards that can be used to post general messages to all users, nad 4) online journals or newsletter.

With computer teleconferencing, meetings can take place economically while allowing users to respond at their leisure. A user usually can communicate more efficiently and effectively. It is self-pacing and requires no performing skills, other than expressing oneself through written words. Computer teleconferencing allows for one to participate in many conferences at the same time.

Ideas differ on whether typing skills (or the lack of them) are a major factor for users of computer teleconferencing systems. It is believed by some that at first a user may be hesitant -- but this is quickly overcome. A user entering comments into the discussion is almost always slower than speaking. However, with computer teleconferencing many users can be "speaking" at the same time -- unlike other forms of teleconferencing.

Several drawbacks of computer teleconferencing exist. First of all, it does not allow, at present, for the presentation of pictures or images, just "typed" diagrams. It also may be difficult to focus the discussion in a computer conference. Some users may be reluctant to put their comments "in writing." Conversely the amount of information entering the conference by everyone can be overwhelming.

2.5.3 Example Uses of Computer Teleconferencing Procter & Gamble employees have been using Confer, a computer teleconferencing system, since 1978. Some conferences go on for months and involve many different managers in the company [13].

The Western Behavioral Sciences Institute in La Jolla, California, is using computer teleconferencing in their WBSI School of Management and Strategic Studies program [5]. This program, using a multimedia approach, gathers executives twice yearly for an eight-day intensive seminar. For the rest oı the time they keep in touch by using several forms of communication including computer teleconferencing.

2.6 <u>About Teleconferencing</u>

This section introduces some issues about teleconferencing in general. Many of these issues are left open because they truly are open issues.

2.6.1 <u>A Substitution For Travel?</u> Should teleconferencing be considered a replacement for face-to-face meetings? Although many teleconferencing system purchases are justified by planning to reduce travel expenses, it may be more important to look at its effects on business communication.

Teleconferencing can change the way most organizations conduct business. In many businesses, meetings are more common than getting work done. Teleconferencing can allow continuous communication resulting in a better business environment. Changing the habits of some people, however, can be a major chore.

2.6.2 <u>Requirements and Needs</u> In order for a teleconferencing system to be successfully introduced into any organization, a clear set of requirements should be developed. These requirements need to specify what is to get done by the conferencing system.

Following the requirements, an organization must look at the specific needs in a teleconferencing system. This process involves looking at the features available in different teleconferencing systems, the capabilities oı each, and the overall costs.

Installing a system will not guarantee its use. By assessing the needs of the business first, it is more likely that the system will be a success.

2.6.3 Gaining the Competitive Advantage Companies that invest in teleconferencing may be doing so for one simple reason. That is, to gain the competitive advantage. With teleconferencing an organization may see more productive employees, reduced travel expenses, and better responsiveness to market opportunities [11]. Although the reduction of travel expenses is the easiest of all to see on paper, the other factors may weigh more in maintaining the company's competitive advantage.

2.7 Applications to Software Development

Computer teleconferencing is especially suitable to R&D and related areas. The stages of a software project beginning with the requirements phase can be streamlined with computer conferencing. Since computer conferencing is self-documenting and because it usually limits itself to relevant discussion, the project members' time may be used more efficiently.

There would usually be no fear of a computer teleconferencing system among members of a software project. In fact it would be simplest and very convenient to work at a computer terminal at one's chosen time, thus reducing scheduling conflicts.

Design reviews can equally be made more efficient with the teleconferencing system. Code reviews can also be adapted to this form of teleconferencing. With code reviews the source code becomes the "document" to be commented on by others. In general, communication would increase, productivity would jump, design flaws would be caught more often, documentation would be better, all without causing inconvenience for project members.

3. Master's Project Implementation Work

3.1 Introduction

This third chapter functionally describes the design of an asynchronous computer teleconferencing system, MARRS, that can be used for the process oi collecting, reviewing and presenting of Master's projects.

The overall objective of this project is to create a computer teleconferencing system applicable to overseeing a portion of design-type projects. A "project leader" will be able to monitor progress, material will be better organized and the system will be self-documenting.

3.2 Overview

The application that the conferencing system will be used for is to oversee the completion of a software design project as functionally shown in Figure 1. The conferencing system is useful in the phases of a software design after it has been committed and funded. The conferencing system oversees the completion of individual assignments of the design project.

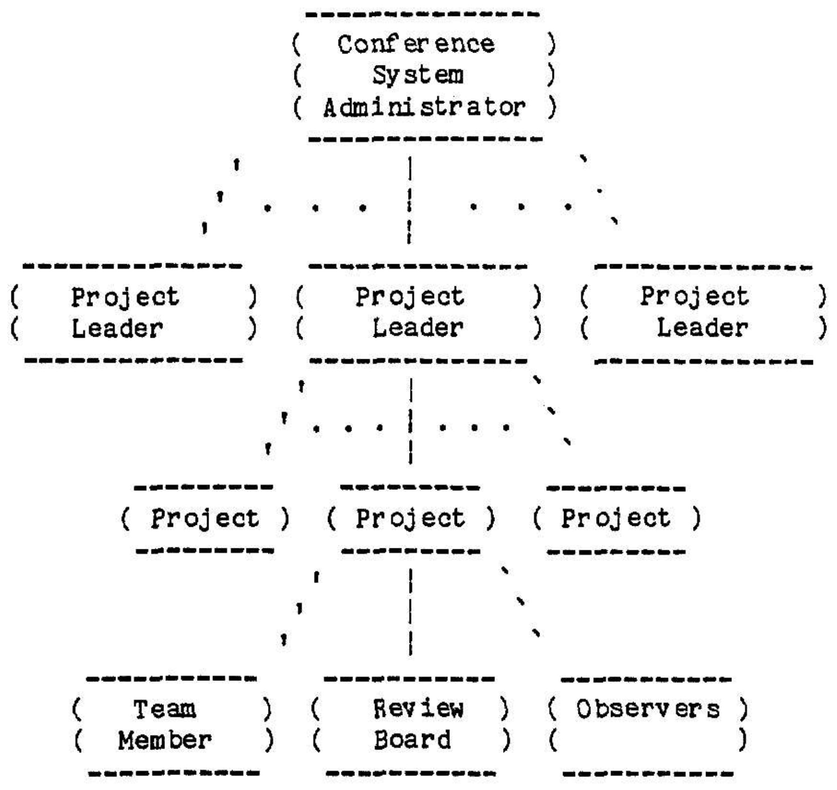

Figure 1. Hierarchy of the users within the final stages of the software design process

In order to ensure that the conferencing system would be useful, it is designed so that it can used for automating the process of collecting, reviewing and presenting of Master's projects. This application is functionally shown in figure 2.

```
                     ---------------
                    (  Conference   )
                    (    System     )
                    ( Administrator )
                     ---------------
                 '          |          `
               '    . . .   |   . . .    `
             '              |              `
   ------------      ------------      ------------
  (   Major    )    (   Major    )    (   Major    )
  ( Professor  )    ( Professor  )    ( Professor  )
   ------------      ------------      ------------
                   '       |       `
                 '  . . .  |  . . .  `
               '           |           `
       ---------      ---------      ---------
      ( Masters )    ( Masters )    ( Masters )
      ( Project )    ( Project )    ( Project )
       ---------      ---------      ---------
                    '      |      `
                  '        |        `
                '          |          `
   -----------      -----------      ----------
  ( Author(s) )    ( Committee )    ( Audience )
   -----------      -----------      ----------
```

Figure 2. Hierarchy of the users in MARRS

The application of this design to the Master's projects involves the following personnel and their roles.

The "conference system administrator" is the overseer of the entire teleconferencing system. The "major professor" is created as the main figure for each conference, or project, in the system. It is the function of the "major professor" to initiate a "masters project." The "major professor" can be associated with one or more "masters projects." It is the function of the "masters project" to contain the author(s), the committee, and the audience of the project.

The remainder of this chapter describes in more detail the functions of the conference system administrator, the major professor, an author, a committee member, and an audience member, as they pertain to the design of this teleconferencing system.

During the creation of specifications and into the design stage, the following assumptions have been made about this project:

1. The selection of committee members is not an application that needs to be part of this system,

2. The end product will run on the Kansas State University VAX system, which is operating with Berkeley UNIX* 4.2,

3. It is not a requirement of the design or coding of this project that the final product be portable,

4. The end product will be accessible to the public,

5. An individual VAX user login name is mandatory in order to use the conferencing system,

6. The project is geared toward providing a "user friendly" environment.

* UNIX is a registered trademark of AT&T Bell Laboratories

3.2.1 Definitions

- Teleconferencing system - A system, either synchronous or asynchronous, that provides interactive group communication through any electronic medium.

- Asynchronous teleconferencing system - A "store and forward" teleconferencing system in which the participants need not all be present at the same time. The electronic medium is usually a computer where the information can be stored and retrieved at the convenience of the user.

- Synchronous teleconferencing system - A "real-time" teleconferencing system in which the participants need be present at the same time, although not in the same location. The electronic medium is usually a video or an audio system.

- Conference system administrator - The initiator of the system. The conference system administrator is responsible for the overall maintenance and performance of the system.

- Major professor - The coordinator of a master's project. The major professor creates a conference which corresponds to a master's project.

- Author - The writer of the master's project paper. An author's paper must be part of a specific conference.

- Committee member - A committee member is the person who is a judge of a paper submitted by an author. Committee member(s) are part of every conference.

- Audience - The participants of a conference, other than the major professor, author(s), and committee member(s).

- Permissions - The protection features of the operating system and the teleconferencing system which are used to determine if a user can access commands or files within a conference. In most cases, a user's login name is used for this purpose.

3.3 Conference Administrator

The conference system administrator oversees the maintenance and performance of the system. The conference administrator is responsible for 1) initiating the conferencing system, 2) establishing a list of faculty members (the professor list) and, 3) maintaining and upgrading the teleconferencing system.

To initiate the conferencing system, the system administrator must set up the "home" directory for the conferencing system. Once established, a source directory is created and the conferencing system's source code and makefiles are read into the "source" directory before the "make" is executed.

Under the MARRS "bin" directory, executable commands and the "professor file" are stored. The "source" directory contains the "makefile", source code, and header files.

The "conference" directory, located under the conferencing system's "home" directory, contains the conferencing directories -- one for each conference. The name or the conference directory is used to identify the conference.

Under each individual conference directory are directories for the author(s), the major professor, the participants. Other files used for the operations of a conference are located here. A "platform" directory, also located under each conference, contains directories for each person in the conference and a "main" directory. The "main" directory contains the master copy of platform discussion files. This files are linked to each user's directory in the "platform" area.

To incorporate security in the system, a list of faculty member login the "professor file", is stored. This file aids in the various permission checks, which vary by each role in the conferencing system.

The conference system administrator classifies each finished report, using the Ingres database system, with the author's name, the date or the report, the major professor, and the keywords of the report. When the final status change is made by the major professor, the system archives the "platrorm" discussion or the report held during the final review.

The conference system administrator can schedule a daily process, a reminder feature, so that deadlines for reviewing or commenting on a paper are not overlooked.

3.4 Major Professor

A professor assumes the role of major professor of a master's project conference by creating the conference. Only a user whose login name is in the professor file can create a conference.

The major professor assigns committee members to the conference. In order to be assigned as a committee member, the individual's login name must appear in the professor file. The major professor is automatically assigned as a committee member. When a major professor assigns a committee member to the conference, the appropriate directory is made and the database and other files are updated.

The major professor assigns the author(s) of the master's project conference. A maximum of ten authors are allowed per conference. An author, in order to be assigned to that role, must have a valid login name. When a major professor assigns an author to the conference, the appropriate directory is made and the database is updated.

The major professor can assign an audience (participants) for the conference. Any user with a valid login name can be made an audience member after requesting access from the major professor of the conference via "mail". When a major professor assigns an audience member to the conference, the appropriate directory is made and the database is updated. An audience member has access to reading presented papers only when the final review procedures have started.

A maximum total of 30 individuals may be in any one conference.

This includes the major professor, committee member(s), author(s), and audience member(s).

The status of a paper is assigned or changed by the major proressor, except in the case when an author causes a status change by submitting a paper to the major professor. (Refer to appendix containing state diagram and description)

The major professor decides on the status of the paper submitted by an author. A status will be one or the following:

1. Paper not submitted,

2. Paper submitted to Major Professor -- awaiting new status,

3. Being rewritten,

4. Ready for committee review,

5. Ready for final review,

6. Paper ready to be published.

The original status is unsubmitted. Only an author is capable of generating status #2, at which time the major professor is notified via "mail." Committee members are notified, via "mail", when status #4 is assigned. All participants in the conference are notified when status #5 is assigned.

Author(s) are notified when any status change occurs on their own paper.

3.5 Author

An author can enter and edit a paper. When appropriate the author can submit the paper to the major professor (status #2).

The author can review comments made by committee members and the audience. These comments can be stored for later access and retrieval. The author can request data on the conference, such as the names of committee members and the audience. The author can query the status of any paper in the conference.

An author is capable of sending information, via "mail", to committee members and the audience.

3.6 Committee Member

The committee member is a faculty member who reviews the paper before its final presentation. A committee member is assigned by the major professor. When a paper goes to the committee review stage, a committee member is notified via computer "mail" of the paper in the conference that is to be reviewed.

The functions or a committee member are to review, comment, and vote on a paper. A committee member is able to review a paper and question an author and/or make on-line comments about the contents of a paper. The teleconferencing system keeps track of the position in the paper so if a committee member wants to temporarily quit reviewing, the current position is marked. A committee member can find out the status of the paper and if the comments that they made have been read.

The comments made by a committee member while the paper is in committee review are accessible only by the author, the major professor, or the committee members. When all committee members have notified the major professor that the paper is ready for final review, the major professor can change the status of the paper from "ready for committee review" to "ready for final review".

At the final review, a committee member votes on the paper from the following choices:

1. Reject it,

2. Major re-work needed,

3. Minor re-work needed, or

4. Acceptable for publication.

The comments made on the final review of the paper can be targeted to certain participants and/or all participants of the conference. Responses to comments targeted to all participants are also available to all participants. This final review stage involves a "platform discussion." The platform discussion differs from the general reviewing procedures in that general discussions can occur which do not relate to specific parts of a paper.

3.7 Audience Member

Any person with a valid system login name can become an audience member by being assigned by the major professor. A list of available conference titles along with the author's name(s) and the associated major professor is available through database queries. This database information can be used in order to request participation in a conference from the major professor. An audience member is notified via "computer mail" once assigned as a participant in the conference by the major professor. The notification tells the audience member the name of the conference that they have been added to.

An audience member can only see and comment on a paper when at the final review state. Audience members are notified via "computer mail" when the status of the paper is "ready for final review." Audience members can review the paper and make on-line comments. The teleconferencing system keeps track of the position in the paper so if an audience member wants to temporarily stop reviewing, the current position is marked. An audience member can find out the status of the paper and if the comments they made have been read.

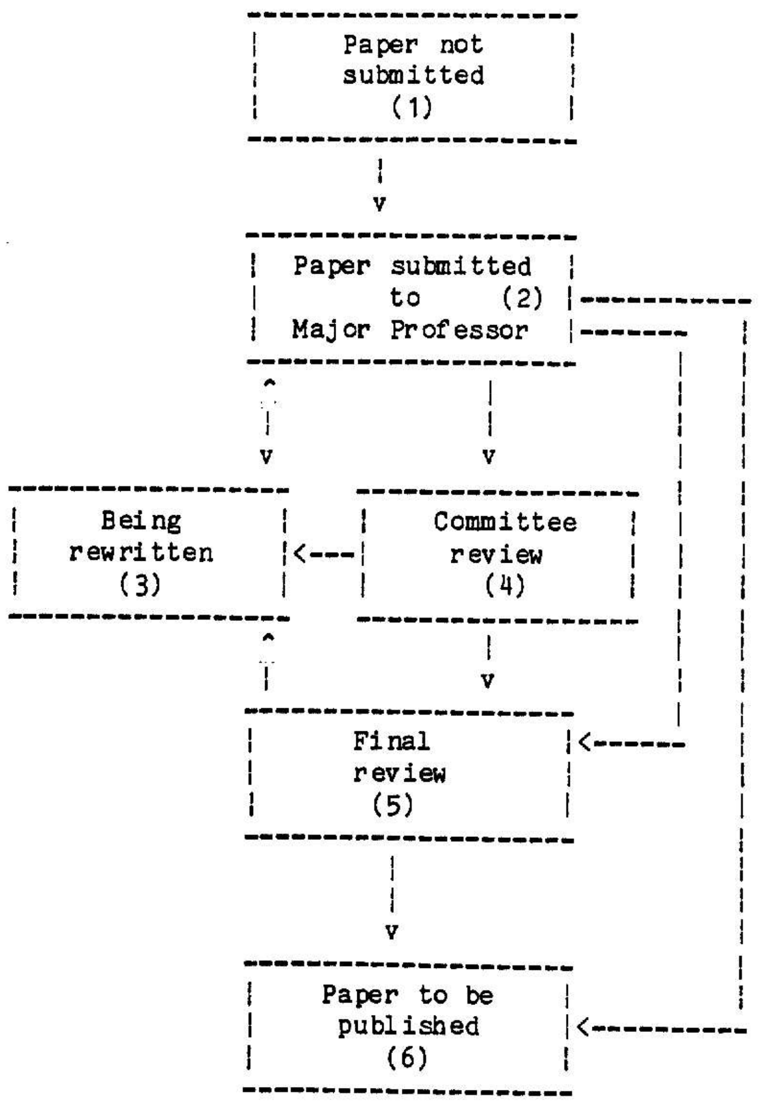

Figure 3. MARRS State Diagram

3.8 State Diagram Description

A paper is initialized to "Paper not submitted" (1). When ready, the author can submit the paper to the Major Professor, causing the state to change to (2). After reviewing the submitted paper, the Major Professor can send it back to the author for revisions (3) or to the committee for review (4). In addition, the paper could be sent to the final review (5) or sent to be published (6) if it is ready and has previously been in states (4) an (5). After being revised (3), an author must submit it back to the Major Professor (2).

After committee review (4) the Major Professor can send it back for revisions (3) or schedule a final review (5). Following the final review the Major Professor can reject it by sending it back to be rewritten (3) or accept it for publication (6).

On the diagram, arrow 1 -> 2 and arrow 3 -> 2 represent state changes due an author's action. All other arrows are a result of a Major Professor's action.

As status cnanges occur for each paper, the time, date, and new status is recorded in a "history" file. This file shows the current status and a concise history of each paper in the conference.

4. Conclusions

4.1 Overview

Teleconferencing can provide many advantages over conventional conferencing depending on the type of teleconferencing selected and how it is used. Teleconferencing can save money by reducing travel expenses, travel time, and wasted conference time. Teleconferencing can allow for more orderly meetings that "get to the point" without wasting an individual's time. However, in some cases, teleconferencing cannot be a valid substitute, especially when face-to-face contact is needed for "social" benefits or for first meetings.

Computer teleconferencing enjoys the added benefit of not requiring conference participants to "meet" simultaneously. It is self-documenting, inexpensive, and allows for a person to "attend" several conferences at one time.

4.2 MARRS

MARRS, MAster's Report Reviewing System, automates the process of collecting, reviewing and presenting of Master's projects. Because it designed as an asynchronous computer teleconferencing system, the participants in the master's report reviewing process need not physically get together for a meeting. The reviewing process is automatically documented by saving the discussions in the conference.

4.3 Future Work

Computer teleconferencing, in general, must address human factors in order to deal with users' hesitations to use a computer. Generally, the non-experienced computer user needs the most "help" in getting used to a computerized teleconferencing system.

The teleconferencing industry must work to reduce costs for video teleconferencing in order for it to be more attractive than audio and audiographics teleconferencing.

Concerning MARRS, several features would enhance its performance. First of all, adding a synchronous "meeting" option to the existing system would allow for real-time meetings. The real-time meeting discussions would be an added feature not intended to replace asynchronous conferencing. The addition of a bulletin board with several category classifications and the ability to add or delete items without an editor would be useful. The addition of a scratchpad area using file encryption for security would allow better assurance that other user's do not view information not intended for them. Comments directed to specific persons, not to be read by others, might also be encrypted.

REFERENCES

1. "An Electronic Bridge Over Troubled Waters", Nation's Business, April, 1984, pp. 78-79.

2. Black, Randall, "Teleconferencing - Send Your Image, Not Yourself", Science Digest, March, 1984, p. 51.

3. Bruce, Harry, "Not Getting There is Half the Fun", Information Technology, March, 1985, pp. 120-126.

4. Cross, Thomas B., "Computer Conferencing", Computerworld on Communications, August 1, 1984, pp. 37-39.

5. Cross, Thomas B., "Computer Tele-conferencing and Education", Educational Technology, April, 1983, pp. 29-31.

6. Cross, Thomas B., "Computer Tele-conferencing - Virtual Networking", Proceedings - Computer Networking Symposium, IEEE Computer Society Press, 1982, pp 3-7.

7. Cross, Thomas B., "The Grapefruit Diet", Journal of Micrographics, April, 1983, pp. 14-18.

8. Dock, Patricia, "Designs and Implementing a Computer Conferencing System to Manage and Track Articles Through the Revision Process", Master's Report, Department of Computer Science, Kansas State University, 1984.

9. Elton, Martin C.J., "Teleconferencing, New Media for Business Meetings", AMA Membership Publications Division, 1982.

10. Goldstein, Mark L.,, "'Your Own CBS', Satellite Video Isn't Videoconferencing", Industry Week, April, 15, 1985, pp. 32-33.

11. Green, David, and Hansell, Kathleen J., "Videoconferencing", Business Horizons, November-December, 1984, pp. 57-61.

12. Heffron, Gordon, "Teleconferencing Comes of Age", IEEE Spectrum, October 1984, pp. 61-66.

13. Johansen, Robert, and Bullen, Christine, "What to Expect From Teleconferencing", Harvard Business Review, March-April, 1984, pp. 164-174.

14. Johansen, Robert, Vallee, Jacques, and Spangler, Kathleen, "Electronic Meetings: Technical Alternatives and Social Choices", Addison-Wesley Publishing Co., 1979.

15. Lazer, Ellen A., and Elton, Martin C.J., Johnson, James W., "The Teleconferencing Handbook", Knowledge Industry Publications Inc., 1983.

16. Olgren, Christine H., Parker, Lorne A., "Teleconferencing: Technology and Applications", Artech House Inc., 1983

17. Parker, Lorne A., and Olgren, Christine H., "Teleconferencing in the United States: Today's Status and Trends", The

Teleconferencing Resource Book: A Guide to Applications and Planning, Lorne A. Parker and Christine H. Olgren (eds.) Elsevier Science Publishers B.V. (North-Holland) IFIP., 1984, pp. 1-13.

18. Panko, Raymond R., "Teleconferencing: A Bridge Across the Pacific?", 1979 Pacific Telecommunications Papers and Proceedings of a Conference held January 8 & 9, 1979, at the Ilikai Hotel, Honolulu, Hawaii, Dan J. Wedemeyer and David L. Jones (eds.) pp. 4E-39 - 4E-45.

19. Pearson, Michael M. L., Kulp, James E., "Creating An Adaptive Computerized Conferencing System On UNIX", Computer Message System, Uhlig, R. P. (editor). North-Holland Publishing Company. IFIP, 1981.

20. "Space Network Readied for McDonnell Douglas", Aviation Week & Space Technology, November 26, 1984, p. 77.

21. Strom, Bernard Ivan, "A Multi-Copy Structured Database Computer Conferencing System", PhD Dissertation, Columbia University, 1980.

22. Strom, B. Ivan, "Computer Conferencing - Past, Present, and Future", Office Information Systems, Naffah, N. (editor). INRIA/North-Holland Publishing Company, 1982.

23. "Systems in Action", Office Administration and Automation,

March, 1984, pp. 82,85.

24. Veith, Richard H., "Television's Teletext", Elsevier Science Publishing Co. Inc. (North-Holland), 1983.

25. "Video Teleconferencing", Aviation Week & Space Technology, December 10, 1984, p. 83.

26. Wolff, Michael F., "What You Should Know About Teleconferencing", Research Management, May-June, 1984, pp. 8-10.

27. Woolfe, Roger, "Videotex", Heyden & Son Ltd, 1980.

Appendix 1

Application Code

```
     /********************************************/
     /*                 bb.c                     */
     /*                                          */
     /*  bb.c is the routine that handles a      */
     /*   conferences bulletin board.  Users     */
     /*   in a conference can read or edit       */
     /*   the conference's bulletin board.       */
     /*                                          */
     /********************************************/


#include <stdio.h>
#include <sys/types.h>          /* for stat call */
#include "stat.h"               /* for stat call */
#include <sys/errno.h>          /* for stat call */

#include "conf.h"

int errno;

char ed_command[SYSTEM_CALL_SIZE];      /* holds system call for editor */
char more_command[SYSTEM_CALL_SIZE];    /* holds system call for "more" */
char Bboard_path[200];          /* path name to Bboard */

char *str, string[20];

main(argc, argv)
int argc;
char *argv[];
{

char login_name[MAX_LOGIN_SIZE];
char conf_path[150];            /* path name to conference */

char conf_name[100];            /* name of conference */

char *getenv(), *editor, conf_editor[MAX_ED_PATH];

        /* First get the user's login name so it can be checked out */

    strcpy(login_name, getlogin());

    printf("Who do you want to execute this routine as? ");
    gets(login_name);

    CLEAR

    printf("                    BULLETIN BOARD\n");

      /* list all conferences that the user is in */
    switch ( usr_conf(login_name, ANY_ROLE, conf_name) )
    {
        case -1:
            return(-1);
            break;

        case 0:
            return(-1);
            break;

        case 1:
            break;

        default :
            fprintf(stderr, "Bad return from routine usr_conf\n");
            return(-1);
            break;
    }

    if(DBUG)
        printf(" - conference is %s\n", conf_name);


    sprintf(Bboard_path, "%s%s.d/%s", CONF_HOME, conf_name, BB_NAME);


    if(( editor = getenv("EDITOR")) == NULL)
        strcpy( conf_editor, DEFAULT_EDITOR );
    else
        strncpy( conf_editor, editor, MAX_ED_PATH - 1);


    sprintf(ed_command,"%s %s", conf_editor, Bboard_path);
    sprintf(more_command,"%s %s", "/usr/ucb/more -cd ", Bboard_path);


    while( TRUE )
    {
        CLEAR
        header();
        str = string;
        gets( str );
        if( routines() ) break;
        sleep(1);
    }
}

header()
{

    printf("\n\n        Valid options are:\n\n");
    printf("                    read   - read the bulletin board\n");
    printf("                    change - change (edit) the bulletin board\n")
    printf("                    quit   - leave the bulletin board\n");
    printf("         Function? : ");
}


routines( )
{

    char outstr[50];

    if (check_input( str, "read") )
    {
        CLEAR

        if (check_BB(Bboard_path) != 0 )
        {
            printf("\nNo information found on bulletin board\n\n");
            sleep(1);
        }
        else
        {
            system(more_command);

            printf("\n\nAT BOTTOM OF BULLETIN BOARD\n");
            printf("HIT RETURN WHEN YOU ARE PREPARED TO CONTINUE ");
            gets( str );
        }

        return( FALSE );
    }

    if (check_input( str, "change") )
    {
        CLEAR
        system(ed_command);
        return( FALSE );
    }



    if (check_input( str, "quit") )
    {
        return( TRUE );
    }

    printf("\nYou entered an invalid string; try again: ");
    sleep(1);
    return( FALSE );
}
```

```
check_BB(file_name)
char *file_name;
{
struct stat *buf, buffer;
extern int errno;

        buf = (&buffer);
        errno = 0;              /* reset in case the is 2nd tiime thru */
        if( (stat(file_name, buf)) == -1 )
            if( errno == ENOENT )       /* file/dir doesn't exist */
                return(-1);
            else
                return(-2);             /* any other reason stat didn't work */
        else
        {       /* found file or directory */

            if( buf->st_size <= 0)      /* File there, but size is zero */
                return (-1);
            else
                 return(0);
        }
}
```

```
     /*********************************************/
     /*                crcon.c                    */
     /*                                           */
     /*  crcon.c is the routine that creates      */
     /*   a conference.  Only a user whose        */
     /*   login name is in the "professor"        */
     /*   file can execute this routine.  The     */
     /*   user then becomes the major professor   */
     /*   of the conference.                      */
     /*                                           */
     /*   This routine prompts for the members    */
     /*   in the conference.                      */
     /*                                           */
     /*********************************************/


#include <stdio.h>
#include <sys/time.h>
#include <sys/file.h>
#include <pwd.h>                  /* for getpwnam call */

#include "conf.h"

int errno;
char conf_name[FILENM_MAX + 1];
char conf_path[150];              /* path name to conference */

int usr_file_index;
int his_file_index;

char login_name[MAX_LOGIN_SIZE];

struct usr_loc_file  usr_loc [ CON_MAX ];
struct stat_his_file stat_his[ PAPER_MAX ];

main(argc, argv)
int argc;
char *argv[];
{

char *getenv();
struct passwd *getpwuid();
int fd, i;

char his_path[100];       /* holds full path to stat_his_file */
char loc_path[100];       /* holds full path to usr_loc_file */

char plat_path[150];              /* holds path to platform directory */


        /* First get the user's login name so it can be checked out */
        /* The person running this program must be in the professor */
        /* file because the person becomes the Major Professor   */

    strcpy(login_name, getlogin());

    switch ( validate_prof(login_name) )
    {
        case -2:
            fprintf(stderr, "ERROR: Could not open professor file\n");
            return(-1);

        case -1:
            printf("Could not find user in professor file\n");
            return(-1);
    }

  /* Must be OK to go on, the person running this is the Major Professor */

    usr_file_index = his_file_index =0;

    CLEAR

    if ( get_conf_name() == -1 )
        return(-1);

    /* Add major professor to the usr_loc_file */
    add_to_usr( MAJ_PROF, usr_file_index + 1, login_name);

    /* Add conference name to data base  */
    add_conf( conf_name, login_name);

    CLEAR

    ask_for_authors(conf_path);

    CLEAR

    ask_for_committee(conf_path, login_name);

    CLEAR

    ask_for_audience(conf_path);

    make_a_dir( conf_path, login_name, "major professor");
    make_a_dir( conf_path, COMMENTS, "comments");

        /* make the platform directory */
    make_a_dir( conf_path, PLATFORM, "platform");

        /* make the main.d under platform directory */
    sprintf(plat_path, "%s/%s", conf_path, PLATFORM);
    make_a_dir( plat_path, "main", "platform_main");

    for(i=0; i<usr_file_index; i++)
    {
        make_a_dir( plat_path, usr_loc[i].usr_id, "platform_user");

        if(usr_loc[i].role == AUTHOR)
            make_a_dir( conf_path, usr_loc[i].usr_id, "author");
    }


    sprintf(loc_path, "%s/%s", conf_path, USR_FILE);
    fd = open( loc_path, O_CREAT|O_WRONLY,0666);

    for(i=0; i < CON_MAX; i++)
    {
        if(usr_loc[i].part_num <= 0)
            break;                          /* done; must be at end */

        write(fd, &usr_loc[i], sizeof(struct usr_loc_file));
    }
    close(fd);


    sprintf(his_path, "%s/%s", conf_path, STAT_FILE);
    fd = open( his_path, O_CREAT|O_WRONLY,0666);

    for(i=0; i < PAPER_MAX; i++)
    {
        if(stat_his[i].art_num <= 0)
            break;                          /* done; must be at end */

        write(fd, &stat_his[i], sizeof(struct stat_his_file));
    }
    close(fd);

}


get_conf_name()
{

char *str, string[100];


int i;
int conf_size, bad_entry, no_conf_name;

    str = string;
```

```
printf("\n                    CREATING A CONFERENCE\n\n");
while ( TRUE )
{
    bad_entry = FALSE;

    printf("Enter the conference name you are the major professor of: ");
    gets(str);

    conf_size = strlen( str );

    if (conf_size == 0) continue;

    if ( conf_size > FILENM_MAX)
    {
            printf("Conference name can be at most %d characters\n",
                                    FILENM_MAX);
            continue;
    }

    strcpy( conf_name, string);

    for (i=0; i<conf_size; i++)
        if(conf_name[i] <= ' ' || conf_name[i] > 'z')
        {
            printf("Sorry, conference name cannot have spaces\n");
            bad_entry = TRUE;
            break;
        }

    if(bad_entry)
        continue;           /* Go ask again */

    bad_entry = -1;         /* re-initialize */

    while (TRUE)
    {
        printf("Conference name entered is '%s', OK? (y or n) ", conf_name
        gets(str);

        switch (string[0])
        {
            case 'y':
            case 'Y':
                    bad_entry = FALSE;
                    break;
            case 'n':
            case 'N':
                    bad_entry = TRUE;
                    break;
        }
```

```
            if(bad_entry == -1)
                continue;       /* Ask again, 'y' or 'n' not given */
            else
                break;
        }

        if(bad_entry == TRUE)
            continue;   /* ask for conf name again, 'n' was entered */

        sprintf(conf_path, "%s%s.d", CONF_HOME, conf_name);

        if ( DBUG )
            printf("%s\n", conf_path);

        errno = 0;              /* reset in case the is 2nd tiime thru */
        no_conf_name = FALSE;
        switch (check_file (conf_path))
        {
            case 0:
                printf("Conference name already exists\n");
                break;

            case -1:
                no_conf_name = TRUE;
                break;

            case -2:
                fprintf(stderr, "Unable to determine status of file: %s\n",
                                                              conf_path);
                perror("");
                break;

        }
        if( no_conf_name == TRUE)
            break;

    }

    if((mkdir(conf_path, CONF_DIR_MODE)) == -1)
    {
        fprintf(stderr, "ERROR: Couldn't make %s conference\n", conf_path);
        perror("");
        return(-1);
    }
    return(0);
}

ask_for_authors(conf)
char conf[];
{
char pap_name[FILENM_MAX + 1];
```

```
int i, pap_size, bad_entry;
int author_num, name_size;

char *str, string[100];
char *pap, paper[100];
char *ans, answer[100];

struct passwd *getpwnam();


    str = string;
    pap = paper;
    ans = answer;
    author_num = 1;

    printf("\nTime to add authors of the conference\n");
    printf("NOTE: All authors must be added at this time!\n");
    while ( TRUE )
    {
        printf("\nEnter login id of author #%d (Enter '.' when done): ",
                                                      author_num);
        gets(str);

        name_size = strlen( str );

        if (name_size == 0)
            continue;

        if( (string[0] == '.') && (name_size == 1))
            return(0);

        if(( getpwnam( str ) == NULL))
        {
           printf("Sorry, author's name must be a valid login ID\n");
           continue;
        }

        if( same_name( str ))
        {
           printf("Sorry, %s is already in the conference\n", string);
           continue;
        }


        while ( TRUE )
        {
            bad_entry = FALSE;

            printf("Enter paper name for %s : ", str);
            gets(pap);
```

```
pap_size = strlen( pap );

if (pap_size == 0) continue;

if (pap_size > FILENM_MAX - 1 )
{
    printf("Paper name can be at most %d characters\n",
                                FILENM_MAX - 1 );
    continue;
 }

strcpy( pap_name, paper);

for (i=0; i<pap_size; i++)
    if(pap_name[i] <= ' ' || pap_name[i] > 'z')
    {
        printf("Sorry, paper name cannot have spaces\n");
        bad_entry = TRUE;
        break;
    }

if(bad_entry)
    continue;                   /* Go ask again */

bad_entry = -1;                 /* re-initialize */

while (TRUE)
{
    printf("Paper name entered is '%s', OK? (y or n) ",
                                          pap_name);
    gets(ans);

    switch (answer[0])
    {
        case 'y':
        case 'Y':
                bad_entry = FALSE;
                break;
        case 'n':
        case 'N':
                bad_entry = TRUE;
                break;
    }

    if(bad_entry == -1)
        continue;   /* Ask again, 'y' or 'n' not given */
    else
        break;
}

if(bad_entry == TRUE)
```

```
                continue;       /* ask again, 'n' was entered */
            else
                break;

        }


        /* Add author to the usr_loc_file */
        add_to_usr( AUTHOR, usr_file_index + 1, str);

        /* Add author to the stat_his_file. Note: usr_file_index was
                                               incremented by 1 in last call*/
        add_to_his( usr_file_index, pap);


        /* Add paper/author to data base */
        add_paper( str, pap_name, conf_name);

        if (++author_num > MAX_AUTHORS)
        {
            printf("Maximum number of authors have been added\n");
            return(0);
        }
    }
}

ask_for_committee(conf, major_prof)
char conf[], *major_prof;
{
char *str, string[100];
char comm_path[150];

int comm_num, name_size;

    str = string;
    comm_num = 2;

    printf("\nTime to add committee members of the conference\n");
    printf("NOTE: All committee must be added at this time!\n\n");

    printf("Committee member '%s' is being added\n", major_prof);

    while ( TRUE )
    {
        printf("Enter login id of committee member #%d (Enter '.' when done):
                                                          comm_num);
        gets(str);
        name_size = strlen( str );

        if (name_size == 0)
            continue;
        if( (string[0] == '.') && (name_size == 1))
            return(0);

        switch ( validate_prof(str) )
        {
            case -2:
                fprintf(stderr, "ERROR: Could not open professor file\n");
                return(-1);

            case -1:
                printf("Could not find '%s' in professor file\n", str);
                continue;
        }

        if( same_name( str ))
        {
           printf("Sorry, %s is already in the conference\n", string);
           continue;
        }

        /* printf("%s", string); */
        comm_num++;

        add_to_usr( COM_MEM, usr_file_index + 1, str);
    }
}

ask_for_audience(conf)
char conf[];
{
char *str, string[100];
char aud_path[150];
struct passwd *getpwnam();

int aud_num, name_size;

    str = string;
    aud_num = 1;

    printf("\nTime to add audience members to this conference\n\n");
    while ( TRUE )
    {
        printf("Enter login id of audience member #%d (Enter '.' when done): ",
                                                        aud_num);
        gets(str);

        name_size = strlen( str );

        if (name_size == 0)
            continue;
        if( (string[0] == '.') && (name_size == 1))
            return(0);

        if(( getpwnam( str ) == NULL))
        {
          printf("Sorry, an audience member's name must be a valid login ID\n"
           continue;
        }

        if( same_name( str ))
        {
           printf("Sorry, %s is already in the conference\n", string);
           continue;
        }
        aud_num++;

        add_to_usr( AUD_MEM, usr_file_index + 1, str);
    }
}

add_to_usr( role, id, name)
int role, id;
char *name;
{
    strncpy(usr_loc[usr_file_index].usr_id, name, LGN_SZ - 1);
    usr_loc[usr_file_index].part_num = id;
    usr_loc[usr_file_index].role = role;

    usr_file_index++;
}

add_to_his( auth_id, paper_name)
int auth_id;
char *paper_name;
{
/* char *ctime(); */

    strncpy(stat_his[his_file_index].paper_file, paper_name, FILENM_MAX );
    stat_his[his_file_index].art_num = auth_id;

    stat_his[his_file_index].status[0].paper_stat = NOT_SUB;
    stat_his[his_file_index].status[0].date = time();

    his_file_index++;
}

same_name(new_name)
char *new_name;
{
int i;
```

```
    if(DBUG)
        printf("usr_file_index is %d\n", usr_file_index);

    for(i=0; i< usr_file_index; i++)
        if(strcmp(usr_loc[i].usr_id, new_name) == NULL)
            return( YES );

    return( NO );
}
```

```
    /*********************************************/
    /*                dump_stat.c                */
    /*                                           */
    /* dump_stat is a tool in the MARRS system   */
    /*  that allows a person to dump the         */
    /*  contents of the "stat_his_file".         */
    /*                                           */
    /* This routine was developed for debugging  */
    /*  purposes and is not included in the      */
    /*  manual as an official part of MARRS.     */
    /*                                           */
    /* It can, however, be useful in locating    */
    /*  problems or for looking at conference    */
    /*  information.                             */
    /*                                           */
    /* It takes a conference name                */
    /*  ON THE COMMAND LINE                      */
    /*                                           */
    /*********************************************/

#include <sys/file.h>
#include <sys/time.h>
#include <stdio.h>
#include "conf.h"


int num_usrs, num_papers;

struct usr_loc_file usr_loc[ CON_MAX ];
struct stat_his_file stat_his[ PAPER_MAX ];


main(argc, argv)
int argc;
char *argv[];
{
    int fdes, i, j;
    char path[MAX_ED_PATH];
    char auth_name[LGN_SZ];
    char text[20];          /* holds status text, e.g. NOT SUBMITTED */
    char *ctime();

    if(DBUG)
        printf("argc is %d\n", argc);

    if(argc == 1)
    {
        printf("Usage: %s conference_name\n", argv[0]);
        exit(-1);
    }
```

```
/* read in the conference's usr_loc_file */

sprintf( path, "%s%s/%s.d/%s", HOME, CONF_DIR, argv[1], USR_FILE );

fdes = open( path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", path);
    exit(-1);
}
if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
{
    CLEAR
    printf("Error in reading %s\n", path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

if (DBUG)
    printf("Number in usr_loc is %d \n", num_usrs);


/* read in the conference's stat_his_file */

sprintf( path, "%s%s/%s.d/%s", HOME, CONF_DIR, argv[1], STAT_FILE );

fdes = open( path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", path);
    exit(-1);
}
if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
{
    CLEAR
    printf("Error in reading %s\n", path );
    close( fdes );
    exit( 0 );
}

printf("\n  ****  Dumping stat_his_file for conference: %s  ****\n\n",
                                                               argv[1]);

num_papers = num_papers / ( sizeof( struct stat_his_file ) );

if (DBUG)
```

```
        printf("Number of papers is %d \n", num_papers);

    for(i=0; i<num_papers; i++)
    {
        get_part_name(&usr_loc[0], stat_his[i].art_num, auth_name);

        printf("\n Author #%d (%s), Paper:
                                                    auth_name,
                                                    stat_his[i].paper_file
        for(j=0; j<STAT_MAX; j++)
        {
            if(stat_his[i].status[j].paper_stat <= 0)
                break;
            else
            {
                printf("Moved to status ");
                get_stat_text( stat_his[i].status[j].paper_stat, text);
                printf("%s", text);
            }

            if(stat_his[i].status[j].date <= 0)
                printf(" on:   BAD DATE\n");
            else
                printf(" on:   %s", ctime(&stat_his[i].status[j].date));


        }

    }

}
```

```
    /**********************************************/
    /*                dump_usr.c                 */
    /*                                           */
    /* dump_usr is a tool in the MARRS system    */
    /*  that allows a person to dump the         */
    /*  contents of the "stat_his_file".         */
    /*                                           */
    /* This routine was developed for debugging */
    /*  purposes and is not included in the      */
    /*  manual as an official part of MARRS.     */
    /*                                           */
    /* It can, however, be useful in locating    */
    /*  problems or for looking at conference    */
    /*  information.                             */
    /*                                           */
    /* It takes a conference name                */
    /*  ON THE COMMAND LINE                      */
    /*                                           */
    /**********************************************/


#include <sys/file.h>
#include <sys/time.h>
#include <stdio.h>
#include "conf.h"


int num_usrs;

struct usr_loc_file usr_loc[ CON_MAX ];


main(argc, argv)
int argc;
char *argv[];
{
    int fdes, i, j;
    char path[MAX_ED_PATH];
    char text[20];

    if(DBUG)
        printf("argc is %d\n", argc);

    if(argc == 1)
    {
        printf("Usage: %s conference_name\n", argv[0]);
        exit(-1);
    }


    /* read in the conference's usr_loc_file */
    sprintf( path, "%s%s/%s.d/%s", HOME, CONF_DIR, argv[1], USR_FILE );

    fdes = open( path, O_RDONLY, 0 );

    if(fdes == EOF)
    {
        fprintf(stderr, "Can't open %s\n", path);
        exit(-1);
    }
    if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
    {
        CLEAR
        printf("Error in reading %s\n", path );
        close( fdes );
        exit( 0 );
    }
    close( fdes );

    num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );
    /* scan all the conference's usr_locs for the login_name */

    if (DBUG)
        printf("Number in usr_loc is %d \n", num_usrs);

    printf("\n  ****  Dumping usr_loc_file for conference: %s  ****\n\n",
                                                                  argv[1]);

    for(i=0; i<num_usrs; i++)
    {
        printf("  %s (ID %d)  --", usr_loc[i].usr_id, usr_loc[i].part_num);

        get_part_text( usr_loc[i].role, text);
        printf("%s\n", text);

        for(j=0; j< PAPER_MAX; j++)
        {
            printf("%d/%d  ", usr_loc[i].location[j].auth_num,
                                 usr_loc[i].location[j].usr_locate);
        }
        printf("\n\n");

    }

}
```

```
     /*********************************************/
     /*               lookcon.c                   */
     /*                                           */
     /* lookcon.c is the routine that lists       */
     /*  information about all available          */
     /*  conferences.                             */
     /*                                           */
     /*********************************************/


#include <sys/file.h>
#include <stdio.h>
#include "conf.h"


FILE *fd, *fopen();


char conf_name[ FILENM_MAX ];
char conf_dir[ FILENM_MAX ];
struct usr_loc_file usr_loc[ CON_MAX ], *user, *maj_prof;
struct stat_his_file stat_his[ PAPER_MAX ], *paper;

main()
{

char str[100];
int op_size, not_finished;

  not_finished = TRUE;

  while (not_finished)
  {

  CLEAR

    header();

    while (TRUE)
    {
        printf("    Enter your desired option: ");
        gets(str);

        op_size = strlen( str );

        if (op_size == 0) continue;


        if (check_input( str, "list") )
        {
            list_all();
    break;
}

if (check_input( str, "major") )
{
    while (TRUE)
    {
        printf("\n    Enter major professor's login name : ");
        gets(str);

        op_size = strlen( str );

        if (op_size != 0) break;
     }

    CLEAR
    l_confmp(str);
    hit_ret();
    break;
}


if (check_input( str, "status") )
{
    CLEAR
    l_confst();
    hit_ret();
    break;
}

if (check_input( str, "author") )
{
    while (TRUE)
    {
        printf("\n    Enter author's login name : ");
        gets(str);

        op_size = strlen( str );

        if (op_size != 0) break;
    }
    CLEAR
    l_confau(str);
    hit_ret();
    break;
}


if (check_input( str, "quit") )
{
    not_finished = FALSE;
```

```
            break;
        }

        printf("\nYou entered an invalid string.\n");
        sleep(1);
    }
  }
}


list_all()
{
    int fdes, i, j, num_papers, num_usrs, tmp_status;

    char path[ MAX_ED_PATH ];
    char author[ 100 ];
    char junk [ 100 ];

    char stat_text[20];



    while(TRUE)
    {
        CLEAR
        printf( "                              CONFERENCE LISTING\n\n" );
        if( list_conf( NULL ) == 0 )
        {
            CLEAR
            printf( "There are no conferences created within marrs.\n" );
            sleep( 2 );
            return( 0 );
        }

        printf("\n");
        while(TRUE)
        {
            printf( "For more information, enter a conference name " );
            printf("('q' to quit): ");

            conf_name[0] = NULL;

            gets( conf_name );

            if((conf_name[0] == NULL) )
                continue;

            if((conf_name[0] == 'q') && (conf_name [1] == NULL) )
                return(0);

            sprintf( conf_dir, "%s.d", conf_name );
        if( list_conf( conf_dir ) )
            break;

        printf("Invalid conference name\n" );
    }

    /* read in the conference's usr_loc_file */

    sprintf( path, "%s%s/%s/%s", HOME, CONF_DIR, conf_dir, USR_FILE );
    fdes = open( path, O_RDONLY, 0 );

    if(fdes == EOF)
    {
        fprintf(stderr, "Can't open %s\n", path);
        close(fdes);
        sleep(2);
        return(-1);
    }

    if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
    {
        printf("Error in the reading usr_loc file of the conference %s.",
                                                              conf_name );
        sleep(2);
        close( fdes );
        return(-1);
    }
    close( fdes );

    num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );


    /* read in the conference's stat_his_file */

    sprintf( path, "%s%s/%s/%s", HOME, CONF_DIR, conf_dir, STAT_FILE );
    fdes = open( path, O_RDONLY, 0 );

    if(fdes == EOF)
    {
        fprintf(stderr, "Can't open %s\n", path);
        close(fdes);
        sleep(2);
        return(-1);
    }
    if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
    {
        printf("Error in the reading history file of the conference %s.",
                                                              conf_name );
        sleep(2);
        close( fdes );
        return(-1);
        }

        num_papers = num_papers / ( sizeof( struct stat_his_file ) );

        CLEAR

        printf("\n                          Conference: %s\n\n", conf_name);
        printf("        AUTHOR          PAPER NAME       PAPER STATUS\n");
        printf("        ------          ----------       ------------\n");

        for(i=0; i < num_papers; i++)
        {
             tmp_status = 0;
             get_part_name( &usr_loc[0], stat_his[i].art_num, author);

             for(j = (STAT_MAX - 1); j >= 0; j--)
                 if( stat_his[i].status[j].paper_stat != 0 )
                 {
                     tmp_status = stat_his[i].status[j].paper_stat;
                     break;
                 }

             get_stat_text( tmp_status, stat_text);

             printf("        %-14s%-16s%s\n", author, stat_his[i].paper_file,
                                                stat_text);
        }

        printf("\n\nMAJOR PROFESSOR:    ");
        list_users( MAJ_PROF, num_usrs);

        printf("\n\nCOMMITTEE MEMBERS:  ");
        list_users( COM_MEM, num_usrs);

        printf("\n\nAUDIENCE MEMBERS:   ");
        list_users( AUD_MEM, num_usrs);

        hit_ret();
    }

}

list_users( job, max )
  int job, max;
{
int i, matches;

    matches = 0;

    for(i=0; i < max; i++)
    {
        if( usr_loc[i].role == job)
        {
            if( (matches % 5) == 0 && (matches > 0) )
                printf("\n                    ");

            matches++;

            printf("%-11s", usr_loc[i].usr_id);
        }
    }

}
header()
{

    printf("\n    Valid options are:\n\n");
    printf("            l - list all conferences\n");
    printf("            s - list conferences and their status\n");
    printf("            m - list conferences by major professor\n");
    printf("            a - list conferences by author's name \n");
    printf("            q - quit\n\n");
}

hit_ret()
{
char junk[100];

        printf("\n\nHit return to continue: ");
        gets(junk);
}
```

```
     /*********************************************/
     /*                paper.c                    */
     /*                                           */
     /*  paper.c is the routine that allows an    */
     /*   author in a conference to edit a        */
     /*   paper.  Following the editting, the     */
     /*   user may be given a chance to submit    */
     /*   the paper to the major professor,       */
     /*   if such a state change is allowed at    */
     /*   that time.                              */
     /*                                           */
     /*********************************************/


#include <stdio.h>
#include <sys/time.h>
#include <sys/file.h>
#include <pwd.h>                    /* for getpwnam call */

#include "conf.h"

int errno;

char login_name[MAX_LOGIN_SIZE];

struct usr_loc_file  usr_loc [ CON_MAX ];
struct stat_his_file stat_his[ PAPER_MAX ];

main(argc, argv)
int argc;
char *argv[];
{

long ttime();

char *ctime();
char *getenv();
char *editor,
     conf_editor[MAX_ED_PATH];

struct passwd *getpwuid();

int stat_diag[MAX_STATES + 1][MAX_STATES - 1];
int fdes, i, j, found_auth, found_num;
int num_usrs, num_papers;
int auth_id, auth_index;

char conf_path[150];                /* path name to conference */
char paper_path[150];               /* path name to paper */
char conf_name[100];                /* name of conference */
char ed_command[SYSTEM_CALL_SIZE];       /* holds system call for editor */
char r_pap_path[100];           /* path name to raw (un-nroff'd) paper */
char pap_path[100];             /* path name to nroff'd paper in MP dir */

char paper_name[FILENM_MAX];

char string [100];

char text[20];

char maj_prof[LGN_SZ];

char auth_name[LGN_SZ ];                /* name of author */

        /* First get the user's login name so it can be checked out */

    strcpy(login_name, getlogin());
    printf("\nWho do you want to execute this routine as? ");
    gets(login_name);

    init_state_diag( &stat_diag[0][0] );

    CLEAR


    printf("                        CREATE/EDIT A PAPER\n");

      /* list all conferences for which the user is an AUTHOR in */

    switch ( usr_conf(login_name, AUTHOR, conf_name) )
    {
        case -1:
            return(-1);
            break;

        case 0:
            return(-1);
            break;

        case 1:
            break;

        default :
            fprintf(stderr, "Bad return from routine usr_conf\n");
            return(-1);
            break;
    }

    if(DBUG)
        printf("In main - conference is %s\n", conf_name);
```

```
/* read in the conference's usr_loc_file */

sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, USR_FILE);
fdes = open( conf_path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", conf_path);
    close( fdes );
    exit(-1);
}
if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
{
    printf("Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

if (DBUG)
    printf("Number in usr_loc is %d \n", num_usrs);


/* read in the conference's stat_his_file */

sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, STAT_FILE);
fdes = open( conf_path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", conf_path);
    close( fdes );
    exit(-1);
}
if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
{
    fprintf(stderr, "Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_papers = num_papers / ( sizeof( struct stat_his_file ) );

if (DBUG)
    printf("Number of papers is %d \n", num_papers);


if (num_papers <= 0)
```

```
{
    printf("No papers in conference\n");
    return(-1);
}


found_auth = NO;

for(i=0; i < num_usrs; i++)
{
    if( strcmp(usr_loc[i].usr_id, login_name) == NULL )
    {
        found_auth = YES;
        auth_id = usr_loc[i].part_num;
        if(DBUG)
            printf("Author id is %d\n", auth_id);
    }
    if( usr_loc[i].role ==  MAJ_PROF )
        strcpy(maj_prof, usr_loc[i].usr_id);

}

if (!(found_auth))
{
    printf("Could not find author in usr_loc file\n");
    sleep(2);
    return(-1);
}

found_num = NO;

for(j=0; j < num_papers; j++)
{
    if( stat_his[j].art_num == auth_id )
    {
        found_num = YES;
        auth_index = j;
        break;
    }
}

if (!(found_num))
{
    printf("Could not find author number in stat_his file\n");
    sleep(2);
    return(-1);
}


sprintf(paper_path, "%s%s.d/%s.d/%s", CONF_HOME, conf_name, login_name,
                                      stat_his[auth_index].paper_file);
```

```
if(DBUG)
    printf("%s\n", paper_path);

if(( editor = getenv("EDITOR")) == NULL)
    strcpy( conf_editor, "/bin/ed");
else
    strncpy( conf_editor, editor, MAX_ED_PATH - 1);

CLEAR

sprintf(ed_command,"%s %s", conf_editor, paper_path);
system( ed_command );

/* read in the conference's stat_his_file for WRITING */

sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, STAT_FILE);

fdes = open(conf_path, O_RDWR, 0);

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s for writing\n", conf_path);
    close( fdes );
    exit(-1);
}

if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
{
    fprintf(stderr, "Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}

lseek(fdes, 0, 0);          /* go to start of file */

num_papers = num_papers / ( sizeof( struct stat_his_file ) );


CLEAR

printf("\nConference: %s\n\n", conf_name);
printf("Author        Paper Status          Status Date\n");
printf("------        ------------          -----------\n");

found_auth = NO;

for( i=0; i<num_papers; i++ )
{
            /* get the author's name */
    get_part_name( &usr_loc[0], stat_his[i].art_num, auth_name);
    if( strcmp(auth_name, login_name) == NULL )
    {

        found_auth = YES;

        for(j= (STAT_MAX - 1); j >= 0; j--)
        {

              /* look backwards for current status */

             if(stat_his[i].status[j].paper_stat > 0)
                break;                /* Found the current status */
        }
        if (j == -1)                  /* didn't find a current status */
            j = 0;                    /* set it to zero */

                /* get the status number's text */
        get_stat_text( stat_his[i].status[j].paper_stat, text);


        printf("%-8s      %-16s       %s\n", auth_name, text,
                                     ctime(&stat_his[i].status[j].date));

        if( check_change ( &stat_diag[0][0],
                              stat_his[i].status[j].paper_stat, PAPER_SUB
        {
            if( j < (STAT_MAX - 1) )
                j++;

                    /* store the new status */
            stat_his[i].status[j].paper_stat = PAPER_SUB;

                    /* store the current time */
            stat_his[i].status[j].date = ttime();
        }
        else
        {
            printf("Sorry, cannot change status\n\n");
            sleep(2);
            return( -1);
        }
    }
    if(found_auth)
    {
        strcpy(paper_name, stat_his[i].paper_file);
        break;
    }
}
```

```
            /* make path to paper in Major Professor's directory */
sprintf(pap_path, "%s%s.d/%s.d/%s", CONF_HOME, conf_name, maj_prof,
                                                         paper_name);

            /* make path to paper in author's directory */
sprintf(r_pap_path, "%s%s.d/%s.d/%s", CONF_HOME, conf_name, login_name,
                                                         paper_name);

if(DBUG)
{
    printf("major prof - %s\n", pap_path);
    printf("raw file - %s\n", r_pap_path);
}


while (TRUE)
{
    string[0] = NULL;

    printf("Changing status from %s to SUBMITTED. OK? ('y' or 'n'): ",
                                                         text);
    gets(string);

    if( (string[0] == 'y') && (string [1] == NULL) )
    {

        /* NOW WRITE THE INFORMATION */

        for(i=0; i<num_papers; i++)
        {
            if( write(fdes,&stat_his[i], sizeof(struct stat_his_file)) !=
                                         sizeof(struct stat_his_file) )
            {
                printf("Error writing stat_his file, errno is %d\n",
                                                           errno);
                sleep(2);
                exit(-1);
            }
        }

                /* Sent mail to the appropriate people */
        stat_chg_mail( &usr_loc[0], conf_name, auth_name, PAPER_SUB);

        nroff_move(r_pap_path, pap_path);

        rm_comments( conf_name, auth_id);

        close( fdes );

        break;
        }
        if( (string[0] == 'n') && (string [1] == NULL) )
            break;

    }

}



long ttime()
{
    return( time() );

}




nroff_move( from_file, to_file)
char *from_file, *to_file;
{

char nroff [SYSTEM_CALL_SIZE];
char *getenv(), *nroff_cmd, command[MAX_ED_PATH];

    if(( nroff_cmd = getenv("MARRS_NROFF")) == NULL)
        strcpy( command, DEFAULT_NROFF );
    else
        strncpy( command, nroff_cmd, MAX_ED_PATH - 1);

    sprintf(nroff, "%s %s > %s",  command, from_file, to_file);

    printf("\nnroff'ing file.  Please wait ... \n");
    system(nroff);

}
```

```
     /*********************************************/
     /*                pr_com.c                  */
     /*  pr_com.c is the routine that prints     */
     /*   out a paper with comments that have    */
     /*   been directed by the user.  If no      */
     /*   comments are present (for the user),  */
     /*   the paper is not printed.              */
     /*                                          */
     /*********************************************/

#include <stdio.h>
#include <sys/file.h>

#include "conf.h"

char comment_path[150]; /* full path to comment file */
char paper_path[150];   /* path to paper in MAJOR PROFESSOR's directory */

char sender[MAX_LOGIN_SIZE];  /* name of persom belonging to a comment */
char print_it[SYSTEM_CALL_SIZE];

struct cmnt_hdr comm_hdr;
char comment[ MAX_CMNT ];
struct cmnt_hdr *cur_comm;
struct cmnt_hdr comm_lst[ 300 ]; /* assume max of 300 comments per paper */

int num_cmnts;

char line[100];

char login_name[MAX_LOGIN_SIZE];

struct usr_loc_file  usr_loc [ CON_MAX ];
struct stat_his_file stat_his[ PAPER_MAX ];


main(argc, argv)
int argc;
char *argv[];
{

FILE *fp, *fp_out, *fopen();
char tmp_str[100];

char *getenv(), *printer, conf_printer[MAX_ED_PATH];

int fdes, i, j, ii;
int num_usrs, num_papers, auth_id, my_id, line_in_pap, line_cmnts, found_auth;

char conf_path[150];            /* path name to conference */
char conf_name[100];            /* name of conference */
char conf_dotd[100];            /* name of conference */

char auth_string [100];


char auth_name[LGN_SZ ];                   /* name of author */
char maj_prof[LGN_SZ ];                    /* name of major_prof */
char paper_name[FILENM_MAX];

        /* First get the user's login name so it can be checked out */
        /* The person running this program must be in the professor */
        /* file because the person becomes the Major Professor  */

    strcpy(login_name, getlogin());

    printf("\nWho do you want to execute this routine as? ");
    gets(login_name);


    CLEAR

    printf("                      PRINT COMMENT MODE \n");

      /* list all conferences for which the user is the MAJOR PROFESSOR */

    switch ( usr_conf(login_name, ANY_ROLE, conf_name) )
    {
        case -1:
            return(-1);
            break;

        case 0:
            return(-1);
            break;

        case 1:
            break;

        default :
            fprintf(stderr, "Bad return from routine usr_conf\n");
            return(-1);
            break;
    }

    if(DBUG)
        printf("In main - conference is %s\n", conf_name);


    /* read in the conference's usr_loc_file */

    sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, USR_FILE);
```

```
fdes = open( conf_path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", conf_path);
    close( fdes );
    exit(-1);
}
if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
{
    printf("Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

if (DBUG)
    printf("Number in usr_loc is %d \n", num_usrs);


/* read in the conference's stat_his_file */

sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, STAT_FILE);
fdes = open( conf_path, O_RDONLY, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", conf_path);
    close( fdes );
    exit(-1);
}
if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
{
    fprintf(stderr, "Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_papers = num_papers / ( sizeof( struct stat_his_file ) );

if (DBUG)
    printf("Number of papers is %d \n", num_papers);


if (num_papers <= 0)
{
    printf("No papers in conference\n");
    return(-1);
```

```
}

for( ii=0; ii<num_usrs; ii++ )
{
    if(strcmp(login_name, usr_loc[ii].usr_id) == NULL)
       my_id = usr_loc[ii].part_num;

    if(usr_loc[ii].role == MAJ_PROF)
       strcpy(maj_prof, usr_loc[ii].usr_id);
}

if(( printer = getenv("MARRS_PRTR")) == NULL)
    strcpy( conf_printer, DEFAULT_PRTR );
else
    strncpy( conf_printer, printer, MAX_ED_PATH - 1);

while( TRUE )
{

    CLEAR

    printf("\nConference: %s\n\n", conf_name);
    printf("           Author         Paper Name\n");
    printf("           ------         ----------\n");

    for( i=0; i<num_papers; i++ )
    {

            /* get the author's name */
        get_part_name( &usr_loc[0], stat_his[i].art_num, auth_name);

        printf("           %-8s      %-16s\n", auth_name,
                                 stat_his[i].paper_file);
    }

    while( TRUE )
    {
        found_auth = NO;

        printf("\nEnter author's name whose paper you want comments on ");
        printf(" ('q' to quit): ");

        gets(auth_string);

            /* check if 'q' (quit) was entered */
        if(auth_string[0] == 'q' && auth_string[1] == NULL)
        {
            return(0);
        }
```

```
            /* look through all the authors */
        for( ii=0; ii<num_usrs; ii++ )
        {
            if((strcmp(auth_string, usr_loc[ii].usr_id) == NULL) &&
                                    (usr_loc[ii].role == AUTHOR))
            {
                found_auth = YES;

                auth_id = usr_loc[ii].part_num;
                break;
            }
        }


        if (found_auth)
            break;          /* good author, go on */
        else
            printf("Invalid author's name\n");
    }

    for( i=0; i<num_papers; i++ )
    {

        if(stat_his[i].art_num == auth_id)
            strcpy(paper_name, stat_his[i].paper_file);
    }


    if (DBUG)
        printf("t%02d.%02d, MP is %s, pname is %s\n", my_id, auth_id,
                                                  maj_prof, paper_name);


    sprintf(comment_path, "%s%s.d/%s/t%02d.%02d", CONF_HOME, conf_name,
                                              COMMENTS, my_id, auth_id);

    sprintf(paper_path, "%s%s.d/%s.d/%s", CONF_HOME, conf_name, maj_prof,
                                                         paper_name);

    if (DBUG)
    {
        printf("path to comments: %s\n", comment_path);
        printf("path to paper: %s\n", paper_path);
    }

    if(check_file(paper_path) != 0)
    {
        printf("Could not find author's paper ");
        printf("in major professor's directory\n");
        sleep (3);
        continue;
```

```
}

if(check_file(comment_path) != 0)
{
    printf("No comments directed to you for that paper.\n");
    sleep (3);
    continue;
}

sprintf(conf_dotd, "%s.d", conf_name);

gath_comm( conf_dotd, my_id, auth_id);

if((fp = fopen( paper_path, "r")) == NULL)
{
    printf("Can't open paper for reading: %s\n", paper_path);
    sleep(3);
    continue;
}

sprintf(tmp_str, "%s%s.%d", "/tmp/marrs.", auth_string, getpid() );

if((fp_out = fopen(tmp_str, "w")) == NULL)
{
    printf("Can't open tmp file for writing: %s\n", tmp_str);
    sleep(3);
    continue;
}

line_in_pap = 1;

while(fgets(line, 90, fp) != NULL )
{
    fprintf( fp_out, "%s", line);     /* put out the line from paper */
    line_cmnts = com_on_line(line_in_pap); /*any comments on the line*,

    for(j=0; j < line_cmnts; j++)
    {
        read_comm(line_in_pap, j);
        fprintf(fp_out,"------- Comment from %-10s -----------\n",
                                                          sender);
        fprintf(fp_out, "%s", comment);
        fprintf(fp_out,"-------------------------------------------\n\r
    }
    line_in_pap++;

    if(line_in_pap >= 40) /*TEMPORARY */
        break;
}
fclose(fp_out);
```

```
        sprintf(print_it, "/bin/pr %s | %s", tmp_str, conf_printer);

/*
        printf("%s\n", print_it);
printf("SYSTEM CALL COMMENTED OUT\n");
sleep(4);
*/
        system(print_it);
        unlink(tmp_str);
    }
}


gath_comm( conf_dir, me, auth)
char *conf_dir;
int me, auth;
 {
     int is_cmnts;
     int i, fdes;
     char path[100];
     struct cmnt_hdr *cmnts_in;
     void qsort();
     int cmpr();

     if( is_comm( conf_dir, me, auth ) )
     {
        cmnts_in = comm_lst;
        num_cmnts = 0;

        fdes = open( comment_path, O_RDWR, 0 );

        for(;;)
        {
           if( read( fdes, cmnts_in, 8 ) != 8 ) break;
           lseek( fdes, cmnts_in->msg_len, 1 );
           cmnts_in++;
           num_cmnts++;
        }
        close( fdes );
        cmnts_in->line_num = -1;
        qsort( comm_lst, num_cmnts, 8, cmpr );
     }
 }

int cmpr( p1, p2 )
struct cmnt_hdr *p1, *p2;
{
    int ret;

    ret = 0;
    if( p1->line_num < p2->line_num ) ret = -1;
    if( p1->line_num > p2->line_num ) ret =  1;
    return( ret );
}


read_comm( for_line, occur )
int for_line, occur;
{
    char path[100];
    int fdes, i;

    comment[0] = NULL;

    fdes = open( comment_path, O_RDWR, 0 );
    for(;;)
    {
        if( read( fdes, &comm_hdr, 8 ) != 8 ) break;
        if( comm_hdr.line_num != for_line )
        {
            lseek( fdes, comm_hdr.msg_len, 1 );
            continue;
        }
        if( occur > 0 )
        {
            occur--;
            lseek( fdes, comm_hdr.msg_len, 1 );
            continue;
        }
        read( fdes, comment, comm_hdr.msg_len );
        comment[ comm_hdr.msg_len ] = NULL;
        get_part_name(&usr_loc[0], comm_hdr.from_id, sender);
        break;
    }
    close( fdes );

}


com_on_line(line)
int line;
{
int i, count;

    count = 0;

    for(i=0; i<num_cmnts; i++)
    {
/*
printf("line is %d, header line is %d\n", line, comm_lst[i].line_num);
*/
        if( comm_lst[i].line_num == line)
```

```
                count++;
            if( comm_lst[i].line_num > line)
                break;
        }
        return(count);
}
```

```
    /*******************************************/
    /*              status.c                  */
    /*                                         */
    /*  status.c is the routine that changes  */
    /*   the status of a paper.  Only the major */
    /*   professor of a conference can change */
    /*   the status of a paper in that        */
    /*   conference.  Only certain status     */
    /*   are permitted from each state.       */
    /*                                         */
    /*******************************************/

#include <stdio.h>
#include <sys/time.h>
#include <sys/file.h>
#include <pwd.h>                /* for getpwnam call */

#include "conf.h"

struct my_stat_file {
    char auth_nm[LGN_SZ ];
    char cur_stat;
    int index;
    int a_index;
    };


char login_name[MAX_LOGIN_SIZE];

struct usr_loc_file  usr_loc [ CON_MAX ];
struct stat_his_file stat_his[ PAPER_MAX ];
struct loc_file *usr_pos;

struct my_stat_file my_stat[ PAPER_MAX ];

main(argc, argv)
int argc;
char *argv[];
{

long ttime();

char *ctime();
struct passwd *getpwuid();


int fdes, i, j, ii;
int num_usrs, num_papers, paper_num, num_publ;
int stat_diag[MAX_STATES + 1][MAX_STATES - 1];
int found_auth,
    auth_chg_only,      /* flag to indicate that author must submit paper */
    good_stat;

short new_stat;

char conf_path[150];              /* path name to conference */
char conf_name[100];              /* name of conference */
char paper_name[FILENM_MAX];              /* name of paper */

char auth_string [100];
char stat [100];

char text[20];

char auth_name[LGN_SZ ];                  /* name of author */

        /* First get the user's login name so it can be checked out */
        /* The person running this program must be in the professor */
        /* file because the person becomes the Major Professor  */

    strcpy(login_name, getlogin());

    printf("\nWho do you want to execute this routine as? ");
    gets(login_name);

    init_state_diag( &stat_diag[0][0] );

    switch ( validate_prof(login_name) )
    {
        case -2:
            fprintf(stderr, "ERROR: Could not open professor file\n");
            return(-1);

        case -1:
            fprintf(stderr, "Could not find user in professor file\n");
            return(-1);
    }

    CLEAR

    printf("                      PAPER STATUS CHANGE\n");

      /* list all conferences for which the user is the MAJOR PROFESSOR */

    switch ( usr_conf(login_name, MAJ_PROF, conf_name) )
    {
        case -1:
            return(-1);
            break;

        case 0:
            return(-1);
            break;

        case 1:
            break;

        default :
            fprintf(stderr, "Bad return from routine usr_conf\n");
            return(-1);
            break;
    }

    if(DBUG)
        printf("In main - conference is %s\n", conf_name);


    /* read in the conference's usr_loc_file */

    sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, USR_FILE);
    fdes = open( conf_path, O_RDONLY, 0 );

    if(fdes == EOF)
    {
        fprintf(stderr, "Can't open %s\n", conf_path);
        close( fdes );
        exit(-1);
    }
    if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
    {
        printf("Error in reading %s\n", conf_path );
        close( fdes );
        exit( 0 );
    }
    close( fdes );

    num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

    if (DBUG)
        printf("Number in usr_loc is %d \n", num_usrs);


    /* read in the conference's stat_his_file */

    sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, STAT_FILE);
    fdes = open( conf_path, O_RDONLY, 0 );

    if(fdes == EOF)
    {
        fprintf(stderr, "Can't open %s\n", conf_path);
        close( fdes );
        exit(-1);
    }
```

```
if( ( num_papers = read( fdes, stat_his, sizeof( stat_his ) ) ) <= 0 )
{
    fprintf(stderr, "Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}
close( fdes );

num_papers = num_papers / ( sizeof( struct stat_his_file ) );

if (DBUG)
    printf("Number of papers is %d \n", num_papers);


if (num_papers <= 0)
{
    printf("No papers in conference\n");
    return(-1);
}

num_publ = 0;

while( TRUE )
{

    CLEAR

    printf("\nConference: %s\n\n", conf_name);
    printf("Author           Paper Status               Status Date\n");
    printf("------           ------------               -----------\n");

    for( i=0; i<num_papers; i++ )
    {
        for(j= (STAT_MAX - 1); j >= 0; j--)
        {
              /* look backwards for current status */

            if(stat_his[i].status[j].paper_stat > 0)
                break;              /* Found the current status */
        }
        if (j == -1)                /* didn't find a current status */
            j = 0;                  /* set it to zero */

            /* get the author's name */
        get_part_name( &usr_loc[0], stat_his[i].art_num, auth_name);

            /* get the status number's text */
        get_stat_text( stat_his[i].status[j].paper_stat, text);

            /* count the papers that are published */
        if( stat_his[i].status[j].paper_stat == PAPER_PUB)
```

```
            num_publ++;

        /* STORE THE INFO IN LOCAL STRUCTURE */

            /* store the name    */
        strcpy(my_stat[i].auth_nm, auth_name);

            /* store the current status   */
        my_stat[i].cur_stat = stat_his[i].status[j].paper_stat;

            /* store the index to the location of the author    */
        my_stat[i].a_index = i;

            /* store the index to the location of the status    */
        my_stat[i].index = j;


        if(DBUG)
            printf("****%s cur %d ind %d a_ind %d\n", my_stat[i].auth_nm,
                                                    my_stat[i].cur_stat,
                                                    my_stat[i].index,
                                                    my_stat[i].a_index);

        printf("%-8s      %-16s       %s", auth_name, text,
                                         ctime(&stat_his[i].status[j].date));
    }

    while( TRUE )
    {
        found_auth = NO;
        auth_chg_only = NO;

        printf("\nEnter author's name whose status will be changed");
        printf(" ('q' to quit): ");

        gets(auth_string);

            /* check if 'q' (quit) was entered */
        if(auth_string[0] == 'q' && auth_string[1] == NULL)
        {
            return(0);
        }


            /* look through all the authors */
        for( ii=0; ii<num_papers; ii++ )
        {
            if(strcmp(auth_string, my_stat[ii].auth_nm) == NULL)
            {
                found_auth = YES;
```

```
                        /* Major professor can't change status if current */
                        /*  stautus is NOT SUBMITTED or RE-WORK           */
                        /*  (only the author can change it to SUBMITTED   */

                    if(my_stat[ii].cur_stat == NOT_SUB    ||
                                 my_stat[ii].cur_stat == RE_WORK )
                        auth_chg_only = YES;

                        /* If paper is in published state, no change allowed *

                    else if(my_stat[ii].cur_stat == PAPER_PUB )
                        auth_chg_only = PAPER_PUB;

                    break;
                }
            }

                /* DON'T ALTER variable ii yet */

            if(auth_chg_only == PAPER_PUB)
            {
                printf("Sorry, paper for %s is in final state.\n",
                                                    my_stat[ii].auth_nm);
            }
            else if(auth_chg_only == YES)
            {
                printf("Sorry, %s must submit paper ", my_stat[ii].auth_nm);
                printf("before a status change can be made\n");
            }
            else if (found_auth)
                break;          /* good author, go on */
            else
                printf("Invalid author's name\n");
        }

                /* DON'T ALTER variable ii yet */

                /* list the new states available */
        new_states( &stat_diag[0][0], my_stat[ii].cur_stat, auth_string );

                /* get author id */
        paper_num = -1 ;

        for(i=0; i<num_usrs; i++)
            if( strcmp( auth_string, usr_loc[i].usr_id) == NULL)
            {
                paper_num = usr_loc[i].part_num;
                break;
            }
    printf(" paper_num is %d\n", paper_num);
```

```
                /* get name of paper */

        for(i=0; i<num_papers; i++)
{
printf("%d %d %d inloop\n", i, stat_his[i].art_num,  paper_num);
            if( stat_his[i].art_num == paper_num)
            {
printf("matched\n");
                strcpy( paper_name, stat_his[i].paper_file);
                break;
            }
}
printf(" paper_name is %s\n", paper_name);

        while(TRUE)
        {
            good_stat = NO;
            new_stat = -1;

            printf("\nEnter new status ('.' for no change): " );
            gets( stat );

            if(stat[0] == '.' && stat[1] == NULL)
                 break;                          /* no change desired */

            switch(stat[0])
            {
                case  'N':
                case  'n':
                         new_stat = NOT_SUB;
                         good_stat = YES;
                         break;
                case  'S':
                case  's':
                         new_stat = PAPER_SUB;
                         good_stat = YES;
                         break;
                case  'R':
                case  'r':
                         new_stat = RE_WORK;
                         good_stat = YES;
                         break;
                case  'C':
                case  'c':
                         new_stat = COM_REVU;
                         good_stat = YES;
                         break;
                case  'F':
                case  'f':
                         new_stat = FIN_REVU;
                         good_stat = YES;
```

```
                        break;
                case  'P':
                case  'p':
                        new_stat = PAPER_PUB;
                        good_stat = YES;
                        break;
                default:
                        break;
            }

               /* Check if a good staus was entered and if that
                  new status is allowed at this time      */

            if((good_stat == YES) &&
                    (check_change( &stat_diag[0][0],
                                 my_stat[ii].cur_stat, new_stat)))
            {
                if (my_stat[ii].index < (STAT_MAX - 1 )) /*don't overflow it *
                    my_stat[ii].index++;

                /* store the new status */
              stat_his[my_stat[ii].a_index].status[my_stat[ii].index].paper_st
                                                             = new_stat;

                /* store the current time */
                stat_his[my_stat[ii].a_index].status[my_stat[ii].index].date =
                                                              ttime();

                /* NOW WRITE THE INFORMATION */

                sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name,
                                                       STAT_FILE);
                fdes = open(conf_path, O_CREAT|O_WRONLY,0666);

                if(fdes == EOF)
                {
                    fprintf(stderr, "Can't open %s\n", conf_path);
                    sleep(2);
                    close( fdes );
                    exit(-1);
                }

                for(i=0; i<num_papers; i++)
                    write(fdes, &stat_his[i], sizeof(struct stat_his_file));

                close( fdes );
/***************/

                /* read in the conference's usr_loc_file */

                sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, USR_FILE
                fdes = open( conf_path, O_RDWR, 0 );

                if(fdes == EOF)
                {
                    fprintf(stderr, "Can't open %s\n", conf_path);
                    sleep(2);
                    close( fdes );
                    exit(-1);
                    }

                if((num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0
                {
                    printf("Error in reading %s\n", conf_path );
                    close( fdes );
                    sleep(2);
                    exit( 0 );
                }

                num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

                lseek(fdes, 0, 0);              /* go to start of file */

printf("Paper num is %d\n", paper_num);

                for( j = 0; j < num_usrs; j++ )
                {
                    for( usr_pos = &( usr_loc[ j ].location[ 0 ] );
                          usr_pos < &( usr_loc[ j ].location[ PAPER_MAX ] );
                          usr_pos++ )
                        if( usr_pos->auth_num == paper_num ) break;

                    if( usr_pos != &( usr_loc[ j ].location[ PAPER_MAX ] ))
                        usr_pos->usr_locate = 0;
                }


                for(i=0; i<num_usrs; i++)
                write(fdes, &usr_loc[i], sizeof(struct usr_loc_file));

                close( fdes );
/***************/

                    /* Send mail to the appropriate people */
                stat_chg_mail( &usr_loc[0], conf_name, my_stat[ii].auth_nm,
                                                                new_stat);


                if ( new_stat == PAPER_PUB )
                {
                        /* mark the paper as published in the data base */
                    upd_paper(auth_string, conf_name, ttime() );
```

```
                        /* prompt for title of the paper */
                    add_title(paper_name, conf_name);

                        /* prompt and store keywords for the paper */
                    add_topic(paper_name, conf_name);

                        /* check if this the only paper not published */
                    if( (num_papers - 1) == num_publ)
                        upd_status(conf_name);

                }

                if (  new_stat == COM_REVU ||
                      new_stat == FIN_REVU || new_stat == PAPER_PUB )
                    rm_comments(conf_name, paper_num);

                break;
            }
            else
                printf("Invalid status\n");
        }

    }

}


new_states( state, old, auth)
int old, state[MAX_STATES + 1] [MAX_STATES - 1];
char *auth;
{
int i;
char text[30];

    CLEAR

    get_stat_text( old, text);

    printf("\nCurrent status for %s is %s\n\n",auth, text);
    printf("Allowable state changes are to:\n");

    for(i=0; i< MAX_STATES - 1 ; i++)
    {
        if( state[old][i] != -1 )
        {
            get_lc_stat_text( state[old][i], text);
            printf("%s\n", text);
        }
        else
            break;
    }
```

```
}

long ttime()
{
    return( time() );

}
```

```
     /*******************************************/
     /*                  subs.c                 */
     /*  subs.c is a miscellaneous collection   */
     /*   short routines.                       */
     /*                                         */
     /*******************************************/


#include <stdio.h>
#include <sys/types.h>          /* for stat call */
#include "stat.h"               /* for stat call */
#include <sys/errno.h>          /* for stat call */
#include <pwd.h>                /* for getpwnam call */

#include "conf.h"



   /* chec_file checks for the presence of a file */

check_file(file_name)
char *file_name;
{
struct stat *buf, buffer;
extern int errno;

        buf = (&buffer);
        errno = 0;              /* reset in case the is 2nd tiime thru */
        if( (stat(file_name, buf)) == -1 )
            if( errno == ENOENT )       /* file/dir doesn't exist */
                return(-1);
            else
                return(-2);             /* any other reason stat didn't work *
        else
            return(0);                  /* found file or directory */
}




/* validate_prof checks if a name is found in the professor file */


validate_prof(prof_name)
char *prof_name;
{
char prof_logid[ MAX_LOGIN_SIZE ];
FILE *fp, *fopen();

    if(( fp = fopen( PROF_FILE, "r")) == NULL )
```

```
        return(-2);     /* Couldn't open file */

    while((fscanf(fp, "%s", prof_logid)) != EOF)
    {
        /* printf("%s\n", prof_name); */
        /* printf("%s\n", prof_logid); */
        if(strcmp(prof_name, prof_logid) == 0)
                return(0);      /* Found a match */
    }

    return(-1);         /* Couldn't find a match */
}



            /* make_a_dir makes a directory */


make_a_dir(conf, dir_name, what_for)
char conf[], *what_for;
char *dir_name;
{
char dir_path[150];

  if((strcmp(what_for,"platform") == 0) || (strcmp(what_for,"comments") == 0))
        sprintf(dir_path, "%s/%s", conf, dir_name );
    else
        sprintf(dir_path, "%s/%s%s", conf, dir_name, DIR_ENDING);

    /* printf("%s\n", dir_path); */

    if((mkdir(dir_path, CONF_DIR_MODE)) == -1)
    {
        fprintf(stderr,"ERROR: Couldn't make %s directory:\n%s  ",
                                                   what_for, dir_path);
        perror("");
        return(-1);
    }

    return(0);
}



/* check_input is used to see if a string matches against another one */


check_input( user_input, check_against )
char *user_input,
     *check_against;
```

```
{

int i;

    i = 0;

    while ( user_input[i] != NULL)
        if (user_input[i] != check_against[i++])
        {
            return(FALSE);
        }

    if ( (i == 0) && (check_against[i]) != NULL) /* no input was entered */
        return(FALSE);

    return(TRUE);

}


        /************************************************************
         *                     get_part_name                        *
         *  This routine takes a pointer to a usr_loc_file          *
         *   structure & a participant number and returns the       *
         *   the participant name in the string pointer passed      *
         *   to the routine.                                        *
         *
         ************************************************************/


get_part_name(fptr, part_id, part_name)

struct usr_loc_file *fptr;
int part_id;
char *part_name;
{

int i, found_flag;

    found_flag = NO;

    if(DBUG)
    {
        printf("ID is %d\n", part_id);
    }

    for(i=0; i<CON_MAX; i++)
    {
        if(fptr->part_num == part_id)
        {
            if(DBUG)
```

```
                    printf("FOUND\n");

                found_flag = YES;
                strcpy( part_name, fptr->usr_id);
                break;
            }
            fptr++;
        }
        return(found_flag);
}


/* get_stat_text returns the text associated with a status number */


get_stat_text( stat_num, stat_text)
int stat_num;
char *stat_text;
{
        switch( stat_num )
        {
            case NOT_SUB :
                    strcpy(stat_text, "NOT SUBMITTED");
                    break;
            case PAPER_SUB :
                    strcpy(stat_text, "SUBMITTED");
                    break;
            case RE_WORK :
                    strcpy(stat_text, "RE-WORK NEEDED");
                    break;
            case COM_REVU :
                    strcpy(stat_text, "COMMITTEE REVIEW");
                    break;
            case FIN_REVU :
                    strcpy(stat_text, "FINAL REVIEW");
                    break;
            case PAPER_PUB :
                    strcpy(stat_text, "PAPER PUBLISHED");
                    break;
            default:
                    strcpy(stat_text, "UNKNOWN STATUS");
        }

}


/* get_part_text returns the text associated with a role number */
```

```
get_part_text( part_num, part_text)
int part_num;
char *part_text;
{
    switch( part_num )
    {
        case MAJ_PROF :
                strcpy(part_text, "Major Professor");
                break;
        case COM_MEM :
                strcpy(part_text, "Committee Member");
                break;
        case AUTHOR :
                strcpy(part_text, "Author of Paper");
                break;
        case AUD_MEM :
                strcpy(part_text, "Audience Member");
                break;
        default:
                strcpy(part_text, "UNKNOWN Member");
    }

}



/* get_stat_text returns the lower-case text associated with a status number *



get_lc_stat_text( stat_num, stat_text)
int stat_num;
char *stat_text;
{
    switch( stat_num )
    {
        case NOT_SUB :
                strcpy(stat_text, "no (Not Submitted)");
                break;
        case PAPER_SUB :
                strcpy(stat_text, "su (Submitted)");
                break;
        case RE_WORK :
                strcpy(stat_text, "re (Re-work Needed)");
                break;
        case COM_REVU :
                strcpy(stat_text, "co (Committee Review)");
                break;
        case FIN_REVU :
                strcpy(stat_text, "fi (Final Review)");
                break;
        case PAPER_PUB :
                strcpy(stat_text, "pa (Paper Published)");
                break;
    }

}


/* stat_chg_mail sends mail to the appropriate participants after
        a paper's status has changed                                 */

stat_chg_mail(usr_ptr, conf_name, author, new_stat)

 struct usr_loc_file *usr_ptr;
 int new_stat;
 char *conf_name;
 char *author;
{
FILE *tmp_fl;
int i;

char mail_target[300];
char call[400];
char file[50];
char stat_text[20];
char abbr_msg[50];


    mail_target[0] = NULL;

    if (DBUG)
    {
        printf("Conf name is %s\n", conf_name);
        printf("author is %s\n", author);
        printf("new stat is %d\n", new_stat);
    }

    switch( new_stat )
    {
        case NOT_SUB :
        case RE_WORK :

                strcpy(mail_target, author);
                break;


        case PAPER_SUB :

                strcpy(mail_target, author);
```

```
            for(i=0; i<CON_MAX; i++)
            {
                if(usr_ptr->role  == MAJ_PROF )
                {
                    strcat(mail_target, " ");
                    strcat(mail_target, usr_ptr->usr_id);

                    break;
                }
                usr_ptr++;
            }
            break;


    case COM_REVU :

            strcpy(mail_target, author);

            for(i=0; i<CON_MAX; i++)
            {
                if(usr_ptr->role  == COM_MEM )
                {
                    strcat(mail_target, " ");
                    strcat(mail_target, usr_ptr->usr_id);
                }
                usr_ptr++;
            }

            break;


    case FIN_REVU :
    case PAPER_PUB :

            for(i=0; i<CON_MAX; i++)
            {
                if(usr_ptr->role  >= MAJ_PROF )
                {
                    strcat(mail_target, " ");
                    strcat(mail_target, usr_ptr->usr_id);
                }

                usr_ptr++;
            }

            break;


    default:
            fprintf(stderr, " ERROR: UNKNOWN STATUS\n");
}
```

```
    printf("real MAIL TARGET: *%s*", mail_target);
    strcpy(mail_target, "janning");
    printf("                MAIL TARGET: *%s*\n\n", mail_target);


    sprintf( file, "/tmp/.m%05d", getpid() );
    if((tmp_fl = fopen( file, "w" )) == NULL)
    {
        fprintf(stderr, "Can't open file for sending mail: %s\n", file);
        return(-1);
    }


    get_stat_text( new_stat, stat_text);

    sprintf( abbr_msg, "marrs - paper status change", 0);
    fprintf(tmp_fl, "\n                  Conference: %s\n\n", conf_name);
    fprintf(tmp_fl, "Status of paper by '%s' has been changed to: %s\n\n",
                                   author,              stat_text);
    fclose( tmp_fl );

    sprintf( call, "mail -s
    system( call );
    unlink( file );


}



        /* init_stat_diag initializes the state diagram table */

init_state_diag( stat_diag )
int stat_diag[ MAX_STATES + 1][ MAX_STATES - 1];
{
int i, j;

    for (i=0; i<MAX_STATES + 1; i++)
        for (j=0; j<MAX_STATES - 1; j++)
                stat_diag[i][j] = -1;  /* initialize to some non-state value *

    /* The following initializes the state diagram table. For each
        state it tells the states that are reachable from that
        particular state. The array is initialized above to some value
        that is NOT a valid state so that only the assignments made
        below can match a request for a state change.
    */

        /* STATE 1          allowable state change(s)*/
    stat_diag[NOT_SUB] [0] =             PAPER_SUB;
```

```
        /* STATE 2          allowable state change(s)*/
    stat_diag[PAPER_SUB] [0] =              RE_WORK;
    stat_diag[PAPER_SUB] [1] =              COM_REVU;
    stat_diag[PAPER_SUB] [2] =              FIN_REVU;
    stat_diag[PAPER_SUB] [3] =              PAPER_PUB;

        /* STATE 3          allowable state change(s)*/
    stat_diag[RE_WORK] [0] =                PAPER_SUB;

        /* STATE 4          allowable state change(s)*/
    stat_diag[COM_REVU] [0] =              RE_WORK;
    stat_diag[COM_REVU] [1] =              FIN_REVU;

        /* STATE 5          allowable state change(s)*/
    stat_diag[FIN_REVU] [0] =              RE_WORK;
    stat_diag[FIN_REVU] [1] =              PAPER_PUB;

        /* STATE 6          allowable state change(s)*/

        /* Can't go anywhere from state PAPER_PUB  --- THE END ---- */


}



       /* check_change checks to see if a status change is legal */


check_change( stat_diag, current, new)
int stat_diag[ MAX_STATES + 1][ MAX_STATES - 1];
int current, new;
{
int i;

    if( current == new )
        return(NO);

    if( current <= 0 || current > MAX_STATES)
        return(NO);

    if( new <= 0 || new > MAX_STATES)
        return(NO);

    for( i=0; i< MAX_STATES - 1; i++)
        if( stat_diag[current][i] == new)
                return( YES );

    return(NO);
}
```

```
       /* rm_comments removes comments about an author's paper */


rm_comments(conf_name, auth_id)
char *conf_name;
int auth_id;
{

char remove[150];        /* full path to comment file */


     sprintf(remove, "/bin/rm -f %s%s.d/%s/t??.%02d", CONF_HOME, conf_name,
                                             COMMENTS, auth_id);

     if (DBUG)
         printf("Removing: %s\n", remove);

     system(remove);
}
```

```
        /*******************************************/
        /*              updmem.c                   */
        /*  updmem.c is the routines that allows a */
        /*   major professor to add audience       */
        /*   members after a conference has been   */
        /*   created.  Max 30 participants.        */
        /*                                         */
        /*******************************************/


#include <stdio.h>
#include <sys/time.h>
#include <sys/file.h>
#include <pwd.h>                    /* for getpwnam call */

#include "conf.h"

char conf_path[150];                /* path name to conference */
char conf_name[50];                 /* conference name */

int usr_file_index;
int aud_count;
int changed_flag;

char login_name[MAX_LOGIN_SIZE];

struct usr_loc_file  usr_loc [ CON_MAX ];

main(argc, argv)
int argc;
char *argv[];
{

char *getenv();
struct passwd *getpwuid();

int fdes, i, num_usrs;


    changed_flag = NO;

    strcpy(login_name, getlogin());

    printf("\nWho do you want to execute this routine as? ");
    gets(login_name);

    CLEAR

    printf("                    ADDING AUDIENCE MEMBER(S)\n");

      /* list all conferences for which the user is the MAJOR PROFESSOR */
```

```
switch ( usr_conf(login_name, MAJ_PROF, conf_name) )
{
    case -1:
        return(-1);
        break;

    case 0:
        return(-1);
        break;

    case 1:
        break;

    default :
        fprintf(stderr, "Bad return from routine usr_conf\n");
        return(-1);
        break;
}

if(DBUG)
    printf("In main - conference is %s\n", conf_name);


/* read in the conference's usr_loc_file */

sprintf(conf_path, "%s%s.d/%s", CONF_HOME, conf_name, USR_FILE);
fdes = open( conf_path, O_RDWR, 0 );

if(fdes == EOF)
{
    fprintf(stderr, "Can't open %s\n", conf_path);
    close( fdes );
    exit(-1);
}
if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
{
    printf("Error in reading %s\n", conf_path );
    close( fdes );
    exit( 0 );
}

num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );

if (DBUG)
    printf("Number in usr_loc is %d \n", num_usrs);

if(num_usrs >= CON_MAX)
{
    printf("The maximum number of participants ");
    printf("already exist in conference.\n");
    sleep(2);
    }

    usr_file_index = num_usrs;

    aud_count = 0;

    for (i=0; i<num_usrs; i++)
        if(usr_loc[i].role == AUD_MEM)
            aud_count++;



    CLEAR

    ask_for_audience(conf_path);


    lseek(fdes, 0, 0);          /* go to start of file */


    if( changed_flag )
    {
        for(i=0; i < CON_MAX; i++)
        {
            if(usr_loc[i].part_num <= 0)
                break;                  /* done; must be at end */

            write(fdes, &usr_loc[i], sizeof(struct usr_loc_file));
        }
    }
    close( fdes );


}



ask_for_audience()
{
char *str, string[100];
char aud_path[150];
struct passwd *getpwnam();

int  name_size;

    str = string;

    aud_count++;

    printf("\nAdding new audience members to this conference\n\n");
    while ( TRUE )
    {
        if(usr_file_index >= CON_MAX)   /* conference is full */
        {
            printf("Conference is full - ");
            printf("cannot have more than %d participants\n", CON_MAX);
            return(0);
        }

        printf("Enter login id of audience member #%d (Enter '.' when done): ",
                                                            aud_count);
        gets(str);

        name_size = strlen( str );

        if (name_size == 0)
            continue;

        if( (string[0] == '.') && (name_size == 1))
            return(0);

        if(( getpwnam( str ) == NULL))
        {
          printf("Sorry, an audience member's name must be a valid login ID\n");
           continue;
        }

        if( same_name( str ))
        {
           printf("Sorry, %s is already in the conference\n", string);
           continue;
        }

        changed_flag = YES;
        add_to_usr( AUD_MEM, usr_file_index , str);
        aud_count++;
    }
}

add_to_usr( role, id, name)
int role, id;
char *name;
{
    strncpy(usr_loc[usr_file_index].usr_id, name, LGN_SZ - 1);
    usr_loc[usr_file_index].part_num = usr_file_index + 1;
    usr_loc[usr_file_index].role = role;

    usr_file_index++;
}
```

```
same_name(new_name)
char *new_name;
{
int i;

    if(DBUG)
        printf("usr_file_index is %d\n", usr_file_index);

    for(i=0; i< usr_file_index; i++)
        if(strcmp(usr_loc[i].usr_id, new_name) == NULL)
            return( YES );

    return( NO );
}
```

```
#include <sys/types.h>
#include <sys/file.h>
#include <stdio.h>
#include "stat.h"
#include <sys/dir.h>
#include <strings.h>
#include "conf.h"

     /*************************************************
      *                                               *
      * usr_conf is a routine to list the conferences*
      * that the user plays a specific role in.       *
      *                                               *
      * The routine takes a login ID and role,        *
      * displays the conferences that the login ID    *
      * has that specific role in and                 *
      * allows the user to select a conf name         *
      * and passes the conference name selected back *
      * to calling routine.                           *
      *                                               *
      ************************************************/

usr_conf( person, role, fill_in_conf_name )
char *person;
int role;
char *fill_in_conf_name;
{
    DIR *opendir();
    DIR *dirp;

    struct direct *readdir();
    struct direct *dp;
    struct stat buf;
    struct usr_loc_file  usr_loc [ CON_MAX ];

    char usr_conf[20][FILENM_MAX];      /* holds user's conference names */
    char path[100];
    char usr_path[100];
    char string[100];
    char *rindex(),*index();

    int count, num_usrs, fdes, i, match_flag;


    if(DBUG)
        printf("User is %s\n", person);

    count = 0;

    sprintf(path, "%s%s/.", HOME, CONF_DIR );
    dirp = opendir( path );
```

```
for( dp = readdir( dirp ); dp != NULL; dp = readdir( dirp ) )
{
    sprintf( path, "%s%s/%s", HOME, CONF_DIR, dp->d_name );
    stat( path, &buf );

    if((buf.st_mode & S_IFDIR) &&
                    (index(dp->d_name, '.') != &(dp->d_name[0])))
    {
        *( rindex( dp->d_name, '.' ) ) = NULL;

        sprintf(usr_path, "%s/%s", path, USR_FILE);
        if(DBUG)
            printf("%s\n", usr_path);

        fdes = open( usr_path, O_RDONLY, 0 );

        if(fdes == EOF)
        {
            fprintf(stderr, "Can't open %s\n", usr_path);
            close( fdes );
            continue;
        }

        if( ( num_usrs = read( fdes, usr_loc, sizeof( usr_loc ) ) ) <= 0 )
        {
            printf("Error in reading %s\n", usr_path );
            close( fdes );
            continue;
        }

        close( fdes );

        num_usrs = num_usrs / ( sizeof( struct usr_loc_file ) );
        if (DBUG)
            printf("Number in usr_loc is %d \n", num_usrs);

        /* scan all the conference's usr_locs for the user */

        for(i=0; i<num_usrs; i++)
        {
            if( (strcmp (usr_loc[i].usr_id, person ) == NULL) &&
                    ((usr_loc[i].role == role) || (role == ANY_ROLE)))
            {
                if(DBUG)
                    printf("FOUND a USER MATCH\n");

                strcpy( usr_conf[count++], dp->d_name);
                break;
            }
        }
        }
    }
    closedir( dirp );

    if(DBUG)
        printf("Total matches is %d\n", count);

    if(count <= 0)
    {
        printf("\n\n");
        printf("No conferences exist for you\n");
        return(-1);
    }
    else if (count == 1)
    {
        strcpy(fill_in_conf_name, usr_conf[0]);
        if(DBUG)
            printf("Only one conference\n");
        return(1);
    }
    else
    {
        printf("\nConferences available:\n\n");
    }

    for(i=0; i<count; i++)
    {
        printf("%18s", usr_conf[i]);
        if( !( (i+1) % 4 ) )
            printf("\n");
    }
    printf("\n");
    match_flag = NO;
    while (TRUE)
    {

        printf("\nEnter a conference name ('q' to quit): ");
        gets( string );

        if(string[0] == NULL)
            continue;

        if( (string[0] =='q') && (string[1] == NULL))
        {
            if(DBUG)
                printf("GOT quit\n");
            return(0);
        }
        for (i=0; i<count; i++)
        {
            if(strcmp(string, usr_conf[i]) == NULL)
            {
                if(DBUG)
                    printf("MATCHED A CONFERENCE NAME\n");
                match_flag = YES;
                break;
            }
        }
        if( match_flag )
            break;
        else
            printf("Invalid conference name\n");
    }

    if(DBUG)
        printf("going back to calling routine-%s-\n", string);

    strcpy(fill_in_conf_name, string);

    return(1);

}
```

```
cc bb.c libmarrs -o /usrb/att/marrs/Bin/bb

cc crcon.c libmarrs libing -lq -o /usrb/att/marrs/Bin/crcon

cc dump_stat.c libmarrs -o /usrb/att/marrs/Bin/dump_stat
cc dump_usr.c libmarrs -o /usrb/att/marrs/Bin/dump_usr

cc lookcon.c libing -lq libmarrs -o /usrb/att/marrs/Bin/lookcon

cc paper.c libmarrs -o /usrb/att/marrs/Bin/paper

cc pr_com.c libmarrs -o /usrb/att/marrs/Bin/pr_com

cc status.c libmarrs libing -lq -o /usrb/att/marrs/Bin/status

cc updmem.c libmarrs -o /usrb/att/marrs/Bin/updmem
```

Appendix 2

MARRS User's Manual

CONTENTS

LIST OF FIGURES

Master's Report Reviewing System (MARRS)

Ronald M. Janning
Kitty Monk
Thomas J. Stachowicz

Abstract

This document describes an application routine that can be used within a group of reviewers to create and review documents. This particular application has been geared at the process of presenting a Master's Report. The MARRS system allows the user to:

- review and comment on a paper,
- leave messages on a bulletin board,
- participate within a platform discussion,
- generate a hardcopy of the comments and paper,
- do a keyword search of topics from published papers within the system,
- list the current conferences within the system, and
- create and edit a paper.

If the user is added to the professor list, the system allows the user to:

- create a conference,
- update the audience members,
- remind idle users, and
- change the status of a paper.

Acknowledgements

This package would not exist without the guidance we received from Rich McBride.

1. Overview

This package is the result of the authors' implementation of a computer conferencing system. Our goal here is not to create a general document reviewing system, but rather an application geared at the creation and reviewing processes of a Master's Report. However, this package with minor modifications could be used within other applications.

In order to gain access to MARRS the user's PATH variable should contain the paths:

```
/usrb/att/marrs/Bin
/usr/ingres/bin
```

After this path is appended to the user's paths, then type the command: 'marrs'. The user's terminal will then clear and MARRS's main menu display will fill the screen. Since there are somethings that only a major professor can do, the page display could vary according to whether or not the user's login name is found in the professor file.

A user can set three environment variables in order to customize the operations of MARRS. The first of these is the variable 'EDITOR'. To set EDITOR, the following two statements can be put in the user's '.profile' or entered at the shell prompt:

```
EDITOR=/usr/ucb/vi
export EDITOR
```

Instead of '/usr/ucb/vi', any other editor can be specified. The default editor used is '/bin/ed'.

The second of these is the variable 'MARRS_PRTR'. To set MARRS_PRTR, the following two statements can be put in the user's '.profile' or entered at the shell prompt:

```
MARRS_PRTR=lpr
export MARRS_PRTR
```

Instead of 'lpr', any other printer can be specified. The default printer used is 'dorm'.

The last of these variables 'MARRS_NROFF'. To set MARRS_NROFF, the following two statements can be put in the user's '.profile' or entered at the shell prompt:

```
MARRS_NROFF='/usr/bin/nroff -mm -n3'
export MARRS_NROFF
```

Instead of '/usr/bin/nroff -mm -n3', any other nroff command can be specified. The default nroff command used is '/usr/bin/nroff -mm'.

1.1 Terminology

In this document, the following words are used:

- conference - a collection of papers that a major professor presides over and contains author(s), committee member(s), and an audience,

- paper - the base unit within the MARRS system that is created by an author and can be reviewed by participants within the conference,

- status changes - the various steps that a paper within the MARRS system can be in at any particular time. The paper can be in any of the following states:

  1. Not submitted,
  2. Submitted,
  3. Rework,
  4. Committee Review,
  5. Final Review, or
  6. Published,

- major professor - The "owner" and "creator" of a conference,

- committee member - one of the judges that determine whether a paper should be published or reworked,

- author - the "owner" and "creator" of a paper within a conference,

- audience member - a participant within a specific conference that can review a paper that is in final review.

1.2 Ideas of Computer Conferencing

Computer teleconferencing, differs from other forms of teleconferencing in that voice communication is not used. In addition, one's geographic location is does not limit its use.

Teleconferences, in general, can be either synchronous or asynchronous. A synchronous teleconference is one that is conducted in real time -- that is, everyone is present at their own location at the same time. An asynchronous teleconference can be conducted without the restriction of everyone "meeting" at the same time.

Computer teleconferencing can be of either form, although asynchronous teleconferencing is more popular. With an asynchronous computer teleconference, the inherent data storing nature of a computer makes it ideal for such a "store and forward" system. Therefore asynchronous computer teleconferencing is not restricted by location or time.

Synchronous computer teleconferencing uses a keyboard for information entry and can tie together many computer users. The dialogue of a user can be seen by the intended recipients as it is entered at the keyboard.

Computer teleconferencing's biggest advantage is that it overcomes geographic and time constraints. This type of teleconferencing is self-documenting so meeting notes or minutes do not have to be documented or reinterpreted.

1.3 Features

The remainder of the sections within this manual will discuss the use of the system. It is divided into various options that are accessible from the main menu page.

The MARRS system allows the user to:

- review and comment on a paper (option - b),
- participate in a platform discussion (option - d),
- generate a hard copy (option - g),
- do a keyword search of topics from published papers within the system (option - k),
- list the current conferences (option - l),
- leave messages on a bulletin board (option - m), and
- create and edit a paper (option - p).

If the user is added to the professor list, the system also allows the user to:

- create a conference (option - c),
- remind idle users (option - r),
- change the status of a paper (option - s), and
- update the audience members (option - u).

1.4 The Data Manager

The data in MARRS is in UNIX files and the INGRES data base management system. The INGRES data base management system is a relational data base system and allows use to be made of INGRES' query/update facilities. File processing is used during the active stages of the conference.

The INGRES data base contains information about the conferences such as conferences title, status, paper titles , keywords, etc. Some of the query/update function that a participant of a conference can perform are :

1. list all the conferences available,
2. find the current status of a conference,
3. add/delete conferences (if authorized),
4. list participants in a conference, and
5. list a participant's biography.

1.5 Limitations

This system was created to run on the Kansas State University VAX system loaded with Berkeley Unix 4.2. It is not guaranteed that this system will work under any other type of hardware or operating system. The source code for the features discussed herein are attachments to the authors' Master's Report.

2. B - Option 'The browser'

This option gives the user the capability to browse and comment a paper. This option should only be used after the major professor has created a conference.

2.1 Startup

The Marrs system will display to the user a list of available conferences. The following is displayed:

```
The following conferences are available.

        aaaa      bbbb      cccc      ddddd

Enter a conference name:
```

If there are no conferences in the system to open, the system will print: There are no conferences created within marrs, and then return the user back to the main menu display. If the user types in an incorrect conference name, Incorrect input: try again, will be displayed and then the system will reprompt you for a conference name. If, however, the correct conference name was typed in but there is a problem with one of the conference's data files, the following message will be displayed:

```
Error in the creation of the conference aaaa Sorry!!
The system administrator of the conferencing system
will be informed.
```

If there is a bad file within the conference the user will be returned to the main menu display.

2.1.1 Becoming an audience member

The user is assigned a specific role within a conference. This role can be one of four roles: major professor, committee member, author, or audience member. All roles except for audience members must be assigned during the creation of the conference. If a user of MARRS types in a conference name that they are not yet a member of, the following is displayed:

```
You do not yet exist within conference aaaa.

Do you want to become a member of the conference? (y or n):
```

The user, whom wants to become an audience member, should answer 'y' to the prompt. As a result, the major professor will be informed through UNIX mail. The user, through UNIX mail, will be informed later of the major professor's decision. MARRS then returns the user to the main menu display.

If the user has entered a correct conference name that they are a member of and there is no problem with the data file the system will check to see if there are any papers in the conference. If there are no papers the system will display: There are no papers created within conference aaaa, and will return the user to the main menu display. On the other hand, if there are papers in the conference the following is displayed:

```
The following papers are available within conference aaaa

        mmmmm     nnnnn     ooooo     pppppp

Enter a paper name:
```

If the incorrect paper name is entered then the system will display: Incorrect input: try again, and will reprompt for the paper name.

2.1.2 Incorrect paper's status?
The paper's status will be checked to assure the condition of the paper matches with the role of the user. For example; if a user is an audience member the paper can only be reviewed during the final review. If a paper is not in the correct state for reviewing and a reviewer attempts to browse and comment the paper, MARRS will inform the user with one of the following error messages:

```
The paper is not yet submitted.
The paper is in the submitted status.
The paper is in the rework status.
The paper is in the committee review status.
The paper has been published.
The system is unable at this time to browse the paper. Sorry!!
```

The user is returned to the main menu display.

2.2 The Page

If the user has been able to make it through all the preliminary setup steps the following is displayed:

```
Would you like to pick up where you left off from? ( y or n )
```

This is only if the user has previously been reviewing the paper. By entering a 'y' the user is returned to the spot where they left off at, any other response returns them to the beginning of the paper.

The browser uses two pages for its displays. The first page displays the paper. The second display is a page where comments can be viewed or entered. Figures A2-(1 thru 5) show the various states of the page displays. WARNING: If some unexpected error happens and the user is kicked out the browser, it is possible that the terminal setting will be incorrect. Enter 'reset' at the unix shell level to reset the terminal settings.

The symbols '>' (greater than) and '<' (less than) as shown in figure A2-1 point to the current line.

```
|-------------------------------------------------------------|
|                                                             |
|                                                             |
|             "Writing is the more personal form              |
|           of communication, the one which permits           |
>           the most natural expression of feeling.           <
|            The message, once detached, can cross            |
|            time and space, acquiring objectivity,           |
|                  permanence and mobility."                  |
|                                                             |
|                      by Andrew Feeberg                      |
|            Western Behavioral Sciences Institute            |
|                                                             |
|-------------------------------------------------------------|
These options are available: c, p, l, q, ?  current line = 5
```

Figure A2-1. The browser page display

The current line is where a comment would be attached if the user were to enter the comment mode to input a comment. When a comment has been attached to the current line, the '<' will be replaced with a '*' (star), as shown in figure A2-2. A comment can only be viewed by moving the '*', found in the outer left hand column, to the current line.

2.2.1 Help

As shown in figure A2-1 a '?' can be entered as one of the options. As a result of entering a '?' the main page display will be cleared and the following information will be displayed on the screen:

```
     The options available within this routine are:

              c (comment) - places you into the comment mode.
[[+|-]n]      l (lines)   - advances the display <n> lines.
[[+|-]n]      p (page)    - advances the display <n> pages.
              q (quit)    - exit the browser routine.
              ? (help)    - to display this page.

               Hit return to continue the browser.
```

After reviewing the information the user hits the return key and MARRS returns the user to the main display page.

2.2.2 Going Forwards

The 'p' (page) and 'l' (line) options are to advance the main display forwards 'n' (a number) of pages or lines, respectively. If the user DOES NOT enter a '-' (minus sign) before entering either a 'p' or an 'l' MARRS will step the user forward through the paper. If a '-' is hit and the user decides before entering the 'p' or 'l' that they would rather go forward then a '+' symbol can be used to cancel the '-'. The variable 'n' is constructed of the sequential numbers that the user enters. The following examples show the user's input and the result of the input:

Input	Result
1 5 p	advance 15 pages forward
+ 1 5 l	advance 15 lines forward
- 1 + 6 p	advance 16 pages forward
- 1 2 3 k p	advance 1 page forward

Shown in the last example a 'k' input is an invalid option to the browser, as a result, the previous information put in -123 is cleared out.

2.2.3 Going Backwards

The 'p' (page) and 'l' (line) options are also used to advance the main display backwards 'n' (a number) of pages or lines, respectively. If the user DOES enter a '-' (minus sign) before entering either a 'p' or an 'l' MARRS will step the user backward through the paper. If a '+' is hit and the user decides before entering the 'p' or 'l' that they would rather go backward then a '-' symbol can be used to cancel the '+'. The variable 'n' is constructed of the sequential numbers that the user enters. The following examples show the user's input and the result of the input:

Input	Result
- 1 5 p	advance 15 pages backward
- 1 5 l	advance 15 lines backward
+ 1 - 6 p	advance 16 pages backward
- 1 2 3 k - p	advance 1 page backward

Shown in the last example a 'k' input is an invalid option to the browser, as a result, the previous information put in -123 is cleared out.

2.2.4 Entering the Comments Mode
In order to attach a comment to a specific line or to view a comment from another reviewer the line must be the current line. To enter the comment mode the user uses the 'c' option. The comment menu page display will be placed over the bottom of the browser's main display page.

2.3 Comments Mode

The screen display shown in figure A1-2 shows the condition of a user that has entered the comment mode while a comment had been attached to the line. If no comment had been entered on the current line the '*' would be a '<' and the comment's menu display would not include the 'v' option.

```
|-------------------------------------------------------|
|                                                       |
|                                                       |
|          "Writing is the more personal form           |
|        of communication, the one which permits        |
>        the most natural expression of feeling.        *
|         The message, once detached, can cross         |
.........................................................
.                                                       .
.  You are currently in the Browser's Comment routine.  .
.                                                       .
.   The options available within this routine are:      .
.                                                       .
. i (input) - to attach a comment to the current line.  .
. q (quit)  - to return to the browsing mode.           .
. v (view)  - to view any current line comments.        .
.                                                       .
.             Enter one of the options.                 .
.                                                       .
.........................................................
```

Figure A2-2. The menu page display while in the comment mode

In order to return to the browsing mode type 'q'. To view comments, type 'v'. And in order to input a new comment enter the 'i' option.

2.3.1 Input
A simple editor was chosen for the design of entering comments. Once in the comment's input mode the user can enter eleven (11) lines of 76 characters per line. If the user tries to go past the 76 character spot, the input will be truncated. Once the input for a line is entered and the newline (carriage return) has been hit, the line of text just entered cannot be edited. If a comment is typed in incorrectly, it is recommended that the user not target the comment entered and reenter the comment's input mode. In figure A2-3 the page display shows the process for entering input.

```
|-----------------------------------------------------------|
|                                                           |
|                                                           |
|           "Writing is the more personal form              |
|         of communication, the one which permits           |
>         the most natural expression of feeling.           *
|          The message, once detached, can cross            |
.............................................................
.      Enter a single '.' on a newline to terminate.       .
.                 Erase char is Backspace                   .
.                                                           .
.          this is the area where the user puts the         .
.          comment that they will later target onto         .
.          other participants of the conference.            .
.                                                           .
.............................................................
```

Figure A2-3. The input page display while in the comment mode

The backspace character must be used in order to erase incorrect input. When finished entering the comment, the user must type a '.' followed by a carriage return on a line by itself to terminate the input mode.

2.3.2 Targeting Comments
After the user has entered the comment MARRS will prompt for the target of the comment. The possible targets depend on the status of the paper. In figure A2-4 the example shows the targets of a paper that is in the final review.

```
|-------------------------------------------------------|
|                                                       |
|                                                       |
|         "Writing is the more personal form            |
|        of communication, the one which permits        |
>        the most natural expression of feeling.        *
|         The message, once detached, can cross         |
.........................................................
.                                                       .
.     Who do you wish to target this comment for?       .
.                                                       .
.   * 1 - major professor ( rich )                      .
.     2 - author ( stachowi )                           .
.     3 - committee member ( unger )                    .
.   * 4 - self ( janning )                              .
.     5 - audience                                      .
.     6 - platform discussion                           .
.                                                       .
.  To target the comment enter a number, or 'q' to quit. .
.                                                       .
.........................................................
```

Figure A2-4. The target page display while in the comment mode

As shown in the example the user has typed in a '1' and a '4', which means that the comment will be attached to the major professor's comment file and their own. If the user types in an incorrect target the input can be canceled by retyping in the number. For example, if a '5' is entered and the user decides that the comment should not be sent to all of the audience members after typing in another '5' the '*' will be erased from the display and the comment will not be sent. In the above example, if the user were to type in a '6' the comment would be added to the conference's platform discussion. For more information concerning the platform discussion see the D option. If the user were to type a 'q' the following message would be displayed:

```
Your comment is currently being sent to the targeted person.
```

Only if a number has a '*' by it will the comment be sent. If none of the numbers have a '*' by them and a 'q' is entered the following message is displayed:

```
Since you didn't target the comment no one will see the input
```

The user is returned to the comment's menu display page.

2.3.3 Viewing Comments

If the current line, as shown in figure A2-5 has a '*' in the left hand column the 'v' option is displayed in the comment's menu display page. After entering a 'v', the comment page is cleared and the first comment is displayed.

```
|-------------------------------------------------------|
|                                                       |
|                                                       |
|          "Writing is the more personal form           |
|        of communication, the one which permits        |
>        the most natural expression of feeling.        *
|         The message, once detached, can cross         |
.........................................................
.                                                       .
.                                                       .
.         this is the area where the comment will       .
.         be displayed that was targeted at the         .
.         reviewer.                                     .
.                                                       .
.                                                       .
.  From: self       These options are available: b, f, q  .
.........................................................
```

Figure A2-5. The viewing page display while in the comment mode The reviewer's name, who entered the comment, is displayed on the last line. If the comment is one of several comments then the author is given the b or f option. These options will take the user backward ('b') or forward ('f') through the list of attached comments. The 'q' option is used to return the user to the comment's menu display.

2.3.4 Leaving the Comments Mode

To leave the comment mode the user must return to the comment's menu display and enter a 'q'. The user will then be returned to the browser's page display.

3. C - Option 'Create a conference'

This option allows a user to create or initialize a conference. Only a user whose login name is found in MARRS's "professor file" can create a conference.

3.1 Naming the Conference

After the "c" option is selected from the main menu, the following is displayed:

```
                    CREATING A CONFERENCE

   Enter the conference name you are the major professor of:
```

A conference name can be at most 15 characters long. Spaces/tabs are not allowed as part of the name. After a name is entered, the following is displayed:

```
   Conference name entered is 'aaaa', OK? (y or n)
```

If an 'n' is entered, the previous prompt will be displayed again. When a 'y' is entered, further checking is done on the conference name. If the conference name has already been used, the user will be informed by a message "Conference name already exists" and a conference name will again be prompted for.

When a valid conference name is entered and a 'y' is entered, the authors of this new conference are prompted for.

3.2 Adding Authors

The section describes how the authors and their paper names are added to the conference.

After the new conference is named, the following is displayed:

```
     Time to add authors of the conference
     NOTE: All authors must be added at this time!

     Enter login id of author #1 (Enter '.' when done):
```

The author's login name must be a valid login name on the system. If it is not, the user will be informed of this and will be prompted for again. When a valid author's name (call it 'stachowi') is entered, the next prompt shows:

```
     Enter paper name for stachowi:
```

A paper name with a maximum of 14 characters is allowed. Spaces/tabs are not allowed as part of the name. After a valid paper name (call it 'telesystems') is entered, the following is displayed:

```
     Paper name entered is 'telesystems', OK? (y or n)
```

If an 'n' is entered, the previous prompt will be displayed again. When a valid paper name is entered and a 'y' is entered, another author will be prompted for (as described above).

There can be a maximum of 10 authors per conference. An author's name cannot be the same as any other participant already entered into the conference.

When all authors and paper names have been added, enter a '.' at the author's login id prompt. The committee members of the conference will now be prompted for.

3.3 Adding Committee Members

The section describes how the committee members are added to a conference. Since only a major professor can create a conference -- and since a major professor is always a committee member, the user creating a conference is automatically added as a committee member.

After all the authors are added by the major professor (called 'rich') , the following is displayed:

```
     Time to add committee members of the conference
     NOTE: All committee must be added at this time!

     Committee member 'rich' is being added
     Enter login id of committee member #2
       (Enter '.' when done):
```

The committee member's login name must be found in MARRS's "professor file". If it is not, the user will be informed of this and will be prompted for again.

A committee member's name cannot be the same as any other participant already entered into the conference. When a valid committee member's name is entered, the next committee member is prompted for.

When all committee members have been added, enter a '.' at the committee member prompt. The audience members of the conference will now be prompted for.

3.4 Adding Audience Members

The section describes how the audience members are added to a conference. Note: All audience members do not have to be added at this point. An option "u" is available at the main MARRS menu to update the audience member list.

After all the committee members are added by the major professor, the following is displayed:

```
     Time to add audience members to this conference

     Enter login id of audience member #1
      (Enter '.' when done):
```

The audience member's login name must be a valid login name on the system. If it is not, the user will be informed of this and will be prompted for it again.

An audience member's name cannot be the same as any other participant already entered into the conference. When a valid audience member's name is entered, the next audience member is prompted for.

When all audience members have been added, enter a '.' at the audience member's id prompt.

3.5 Conference Structure Setup

Following the entry of the conference participants, appropriate directories are created and files written. The user is then returned to the main MARRS menu.

The conference setup is done automatically. Therefore the user does not have to take any more specific actions. However, the user should allow several seconds for the program to setup the conference. Once the user is returned to the main MARRS menu, the setup is complete.

4. D - Option 'The (Platform) discussion'

This option gives the user the capability to view or attach an item in the platform discussion. The items in the platform discussion are available to all participants of the conference. There is a restriction in the platform discussion that at least one of the papers in the conference must be in final review. Once in the platform discussion, the user gets a list of conferences that the user is a member of and providing that there is more than one conference, the user will be requested to enter a conference name. The user can enter 'q' to quit at this point and return to the marrs main menu display. The user can also enter a conference name. If the user is a member of only one conference, then the user is not requested to enter conference name. The following screen is display for a user that is a member of more than one conference :

```
Conferences available:

        marrs              tjs1             tjs2
        star_wars          test_con         dummy_con

Enter conference name :
```

The user then enters a conference name from the list of available conferences. If the user enters a conference and there are no papers in final review for that conference, the following message is printed :

```
Sorry, there are no papers in final review for conference
(conference name).
```

If the user enters 'marrs' as the conference name and there is a paper in final review, then the following is displayed :

```
Conference : marrs
Open items :  none

Enter Valid Option ( t, a, q, ? ) :
```

The 'Open items' field lists any new items that the user has not read. Once the user reads an item, it will no longer be listed as an open item. The user can now select one of the available options. They are the following :

- type(t)
- attach(a)
- quit(q)
- help(?)

4.1 Type(t)

The type option allows the user to print (or type) out an item that is in the platform discussion. If there are no items in the platform discussion and the user selects the type option, the following message is printed :

```
Sorry, there are no items in the platform
discussion for conference (conference name).
```

If there are items in the platform discussion, the following is displayed when the 't' option is entered :

```
Conference : marrs
Open items :  none

Enter Valid Option ( t, a, q, ? ) : t

Items available:
 1  2  3

Enter Item Number (q to quit) : 2

Item 2: ( monk )
   response to item: #1
   followed by item: #3

This is platform item 2.



End of item.   Hit return to continue
```

The 'Items available' is a list of all items in the platform discussion. The user enters an item number followed by a carriage return. In this example, a '2' was entered. The heading gives the item number, login name of the person who made the comment, what item this item is in response to, and also if this item is followed by another item. If the item is not in response to another item or if it is not followed by another item, then these lines are not printed. The item is then printed followed by a message that says you're at the end of item and to hit return to continue with the platform discussion.

4.2 Attach(a)

The attach option allows the user to enter a new item in the platform discussion and if desired to attach this item to another item in the platform discussion. An item can only have one successor and one predecessor. If the user tries to attach the item to an item that already has a response to it (successor), the following message is printed :

```
Item already has a response.
```

An example of the attach option follows :

```
Conference : marrs
Open items :  none

Enter Valid Option ( t, a, q, ? ) : a

Response to item # (0 for none) : 0

You are in the 'ed' editor
0
```

The 'Response to item # (0 for none)' line is printed only when there are items in the platform discussion. If the user just wants to enter a new item, a '0' indicates that this in not in response to any other item. The user is then put into an editor. If the user's EDITOR environment variable is set, the user will be put in that editor. If it isn't, then the default editor is the 'ed' editor. When the user finishes entering the new item, the user enters a 'w' to write the new item and then a 'q' (if the 'ed' editor is being used) to quit the editor. The user then returns to the platform discussion display. All other users in this conference will get this new item as an open item when they enter the platform discussion.

4.3 Help(?)

The help (?) option is to help explain to the user what the various options of the platform discussion are. The following screen is displayed :

```
Conference : marrs
Open items :  none

Enter Valid Option ( t, a, q, ? ) :

     Valid options are:

               a - add an item to the platform discussion
               t - type an item in the platform discussion
               ? - this display
               q - leave the platform discussion


     Hit return to continue
```

To continue with the platform discussion, the user hits the return.

4.4 Quit

To exit the platform discussion, the user enters a 'q' for quit. The user returns back to the marrs main menu display. An example is :

```
Conference : marrs
Open items :  none

Enter Valid Option ( t, a, q, ? ) : q
```

The user is returned to then returned to the marrs main menu display.

5. G - Generate a Hard Copy

This option allows a user to generate a hard copy of a paper. The hard copy includes all the comments on the paper that were directed to the user. A user can only use this option in a conference to which he/she belongs.

5.1 Selecting the Conference

After the "g" option is selected from the main menu, a search is done to see what conferences are available in which the user is a participant.

If no conferences exist for which the user is a participant, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is a participant in, the user will skip this part on "Selecting the Conference" and go directly to the "Selecting the Author/Paper" section.

If, however, more than one conference exists for which the user is the major professor of, the following display is shown:

```
                    PRINT COMMENT MODE

     Conferences available:

            aaaa          bbbb          cccc          dddd


     Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is allowed to select the author of the paper to be printed.

5.2 Selecting the Author/Paper

Following the selection of a conference, the next display shows something like:

```
                   Conference: aaaa
```

```
                Author          Paper Name
                ------          ----------
                stachowi        telesystems
                janning         features
                monk            datamanager

    Enter author's name whose paper you want comments on
     ('q' to quit):
```

The author's login name entered must be from the list presented. If it is not, the user will be informed of this and will be prompted for another name.

When a valid audience member's name is entered, a check is done to see if the paper is available for reading. If it is not, the user will get an appropriate message and be prompted for another name.

If no comments have been directed to the user on the specified author/paper, the user will get an appropriate message and be prompted for another name.

If the paper is present and there are comments for the user, the paper and the comments will be spooled to a printer. Note: The printer can be selected by the environment variable MARRS_PRTR -- See the Introduction.

Following this, the author/paper menu is displayed again. At this point, another author's name can be entered. If a 'q' is entered, the user is returned to the main MARRS menu.

6. K - Option 'The Keyword Search'

This option gives the user the capability to perform keyword searching on published papers. A paper that is in the 'Ready to be published' state has keyword topics associated with it. There can be a maximum of ten keywords per paper. When the user enters the keyword search routine, the following is displayed :

```
Enter topic (? for help, q to quit) :
```

The user can enter a topic or a '?' for a help message. If a '?' is entered, the following is displayed :

```
The Keyword search provides the capability to
list the conference names and paper file names
associated with a requested topic.

The topic can be at most 20 characters.


Hit return to continue
```

After the user enters a topic, the following is displayed :

```
Conference: marrs
Paper name: features

Conference: marrs
Paper name: datamanager

Conference: marrs
Paper name: telesystems


Enter paper file to print paper title (0 for none) :
```

The user can now enter the paper name and then will be prompted for the conference name. The user can enter a 'q' to continue or the conference name. If the user enters the conference name the paper's title is printed. A '0' can be entered to continue with the keyword search. If a paper name is entered, say 'datamanager', the following is displayed :

```
Paper title: datamanager

Data Management in MARRS


Hit return to continue
```

7. L - List Current Conferences

This option allows a user to see what conferences currently exist on the system. A user can choose to look at the status of all conferences or choose conferences that have a specific major professor or author. In addition, the user can select specific conference in order to get additional information on the conference.

7.1 Selecting the Listing Option

After the "l" option is selected from the main menu, a new menu is displayed. Note: The use of this option does not require that the user belong to the conference.

When conferences exist on the system, a display similar to the following is shown:

```
    Valid options are:

            l - list all conferences
            s - list conferences and their status
            m - list conferences by major professor
            a - list conferences by author's name
            q - quit

    Enter your desired option:
```

The user can enter any of the options listed. Following completion of that option's routine, control is passed back to the previous menu. At this point the user can quit or enter another option.

7.1.1 The 'l' option - Selecting the Conference

After the "l" option is selected from the main menu, all conference names are displayed. Note: The use of this option does not require that the user belong to the conference.

When conferences exist on the system, a display similar to the following is shown:

```
                          CONFERENCE LISTING

      Conferences available:

             aaaa          bbbb          cccc          dddd
             eeee          ffff

      For more information, enter a conference name
       ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

If a valid conference name is entered, the user is given more information on the conference. The following is an example of what might be printed for conference 'bbbb':

```
                     Conference: bbbb

       AUTHOR          PAPER NAME       PAPER STATUS
       ------          ----------       ------------
       stachowi        telesystems      SUBMITTED
       janning         features         SUBMITTED
       monk            datamanager      FINAL REVIEW


  MAJOR PROFESSOR:    rich

  COMMITTEE MEMBERS:  gustoff  unger

  AUDIENCE MEMBERS:   humphry  hummel   chance   merrit

  Hit return to continue:
```

At this point, a carriage return can be entered to get to the "conference listing" menu previously described. The user can then select another conference name or enter "q" to quit and return to the main MARRS menu.

7.1.2 The 's' option - Selecting the Conference

If an 's' is entered at the option menu, all past and present conferences and their statuses are displayed. The following shows what a typical screen would show.

```
    Conference: aaaaaa          Status: open
    Conference: bbbbbbb         Status: closed
    Conference: ccccc           Status: open
    Conference: dddd            Status: open
    Conference: eeeee           Status: closed


    Hit return to continue:
```

At this point the user can hit a carriage return when finished observing the display. Control is then passed back to the options menu.

7.1.3 The 'm' option - Selecting Conferences by Major Professor

If an 'm' is entered at the option menu, the system will display the following:

```
    Enter major professor's login name :
```

The user then can enter a major professor's login name. The following display is the shown (assume "rich" was entered as the major professor's name):

```
    Conferences with major professor rich:

    aaaaaa          cccccccc        ddddd


    Hit return to continue:
```

At this point the user can hit a carriage return when finished observing the display. Control is then passed back to the options menu.

7.1.4 The 'a' option - Selecting Conferences by Author

If an 'a' is entered at the option menu, the system will display the following:

```
    Enter author's login name :
```

The user then can enter an author's login name. The following display is the shown (assume "rich" was entered as the major professor's name):

```
    Conferences with author monk:

    ffffff          gggggggg        hhhhh


    Hit return to continue:
```

At this point the user can hit a carriage return when finished observing the display. Control is then passed back to the options menu.

8. M - Leave Message on Bulletin Board

This option allows a user to read and/or change the bulletin board for a specific conference. An editor is used to add, delete, or modify the bulletin board. A user can only use this option in a conference to which he/she is a participant.

8.1 Selecting the Conference

After the "m" option is selected from the main menu, a search is done to see what conferences are available in which the user is a participant.

If no conferences exist for which the user is a participant, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is a participant in, the user will skip this part on "Selecting the Conference" and go directly to the "Bulletin Board Option Selection" section.

If, however, more than one conference exists for which the user is a participant in, the following display is shown:

```
                         BULLETIN BOARD

     Conferences available:

               aaaa         bbbb          cccc          dddd

    Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is given the bulletin board option menu.

8.2 Bulletin Board Option Selection

Following the selection of a conference, the user is given a choice of several options as shown:

```
        Valid options are:
```

```
                    read   - read the bulletin board
                    change - change (edit) the bulletin board
                    quit   - leave the bulletin board

        Function? :
```

If the user selects "q" (quit), control is passed back to the main MARRS menu.

8.2.1 Change Bulletin Board
If a "c" is entered at the bulletin board option display, an editor is invoked so that the user can add to, delete from, or modify the conferences bulletin board.

An editor is used to alter the bulletin board. The bulletin board can be made in any desired format or style. Note: The editor used can be selected by the environment variable EDITOR -- See the Introduction.

When the editor is exited, control is passed back to the bulletin board option menu.

8.2.2 Read Bulletin Board
If a "r" is entered at the bulletin board option display, the "more" command is invoked so that the user can read the board. If no information is present on the bulletin board, a message indicating this will be output and control is passed back to the option menu.

If there is information on the bulletin board, it will be displayed. If more than one page is present, the space bar can be used to advance the board by a page. A <cr> (carriage return) can be used to advance by a single line.

When the end of the board is reached, the user will see:

```
     AT BOTTOM OF BULLETIN BOARD
     HIT RETURN WHEN YOU ARE PREPARED TO CONTINUE
```

At this point a <cr> can be entered to get back to the following options menu:

```
        Valid options are:

                    read   - read the bulletin board
                    change - change (edit) the bulletin board
                    quit   - leave the bulletin board
        Function? :
```

Again the user now has the choice of r(eading), c(hanging), or q(uitting).

9. P - Create/Edit a Paper

This option allows a user to edit a paper in a conference. The editor is used to create the paper when it hasn't already been created. A user can only use this option in a conference which he/she is an author.

9.1 Selecting the Conference

After the "p" option is selected from the main menu, a search is done to see what conferences the user is an author in.

If no conferences exist for which the user is an author, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is an author in, the user will skip this part on "Selecting the Conference" and go directly to the "Editting the Paper" section.

If, however, more than one conference exists for which the user is an author in, the following display is shown:

```
                         CREATE/EDIT A PAPER

     Conferences available:

            aaaa          bbbb          cccc          dddd

    Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is allowed to edit his/her paper.

9.2 Editing the Paper

Following the selection of a conference, an editor is invoked so that the user can alter the paper. Note: The editor used can be selected by the environment variable EDITOR -- See the Introduction.

When the editor is exited, the user may be given a chance to submit the paper to the major professor.

9.3 Submitting the Paper

Following the editing session, the user is given a chance to submit the paper if it is in the "not submitted" or the "re-work" state. If this is the case, something like the following is displayed:

```
                    Conference: aaaa

    Author    Paper Status        Status Date
    ------    ------------        -----------
    janning   NOT SUBMITTED       Wed Jun 11 13:33:56 1986

    Changing status from NOT SUBMITTED to SUBMITTED.
      OK? ('y' or 'n'):
```

If the user desires to submit the paper, a 'y' should be entered. At this point, the paper is "nroff'd" and comments associated with the paper are removed. The user is then returned to the main MARRS menu. Note: The nroff command used can be selected by the environment variable MARRS_NROFF -- See the Introduction.

If an 'n' is entered in response to the 'change status' question, the user is simply returned to the main MARRS menu.

If following the editing session, the paper is not in a state where it can be changed to "submitted", information on the paper is shown (as above) along with a message indicating that a status change is not allowed. The user is then returned to the main MARRS menu.

10. R - Reminde Idle Users

This option allows a *major professor* to remind idle users in a conference. A user can only use this option for a conference in which he/she is a major professor.

As a conference proceeds, the papers in that conference undergo several state changes. Two of these states are the committee and final review. This option gives the major professor the ability to remind idle users that a specific author's paper within a conference is being reviewed. An idle reviewer is defined by the condition of a reviewer that has no previous location stored for the paper.

10.1 Selecting the Conference

After the "r" option is selected from the main menu, a search is done to see what conferences are available in which the user is a major professor.

If no conferences exist for which the user is a major professor, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is a major professor in, the user will skip this part on "Selecting the Conference" and go directly to the "Selecting the Author" section.

If, however, more than one conference exists for which the user is the major professor of, the following display is shown:

```
                    REMIND IDLE USERS


   Conferences available:

            aaaa          bbbb          cccc          dddd


   Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is allowed to select an author of the paper whose participants are to be reminded.

10.2 Selecting the Author

Following the selection of a conference, the next display shows something like:

```
                    Conference: cccc

Author         Paper Status        Status Date
------         ------------        -----------
janning        FINAL REVIEW        Thu Jun 11 07:35:59 1986
stachowi       COMMITTEE REVIEW    Thu Jun 10 07:31:58 1986

Enter author's name whose
reviewers are to be reminded, ('q' to quit) :
```

The author's login name entered must be from the list presented. If it is not, the user will be informed of this and will be prompted for another name. Entering a "q" will return the user to the main MARRS menu.

After a valid author's login name is entered the MARRS routine will issue mail to all idle user's of the paper. The user is then reprompted, if there is more than one paper, for another author's name. If there is only one paper or after the user types in a "q" for the author's prompt the user is returned to the main MARRS menu.

11. S - Change Status of Paper'

This option allows a user to change the status of a paper in a conference. A user can only use this option for a conference in which he/she is a major professor.

As a conference proceeds, the papers in that conference undergo several status changes. Only certain changes are allowed from any given state. The following table shows the allowable state changes:

State		Allowable new state(s)
Not submitted	->	paper submitted
paper submitted	->	re-work, committee review final review, paper published
re-work	->	paper submitted
committee review	->	re-work, final review
final review	->	re-work, paper published
paper published	->	(no allowable state changes)

11.1 Selecting the Conference

After the "s" option is selected from the main menu, a search is done to see what conferences are available in which the user is a major professor.

If no conferences exist for which the user is a major professor, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is a major professor in, the user will skip this part on "Selecting the Conference" and go directly to the "Selecting the Author" section.

If, however, more than one conference exists for which the user is the major professor of, the following display is shown:

```
                    PAPER STATUS CHANGE
```

```
     Conferences available:

             aaaa         bbbb         cccc         dddd


     Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is allowed to select an author of the paper whose status will be changed.

11.2 Selecting the Author

Following the selection of a conference, the next display shows something like:

```
                    Conference: cccc

     Author       Paper Status     Status Date
     ------       ------------     -----------
     janning      SUBMITTED        Thu Jun 11 07:35:59 1986
     stachowi     SUBMITTED        Thu Jun 10 07:31:58 1986

     Enter author's name whose status
      will be changed ('q' to quit):
```

The author's login name entered must be from the list presented. If it is not, the user will be informed of this and will be prompted for another name. Entering a "q" will return the user to the main MARRS menu.

When a valid audience member's name is entered, a new display will show the allowable state changes for the author/paper selected.

11.3 Making the Status Change

Following the selection of aa author, the next display shows something like:

```
     Current status for stachowi is SUBMITTED
     Allowable state changes are to:
                            re (Re-work Needed)
                            co (Committee Review)
                            fi (Final Review)
                            pa (Paper Published)

     Enter new status ('.' for no change):
```

The user can now enter any of the available state changes listed on the display. If an invalid state is entered, the user is informed of this and re-prompted for a new status.

If a valid state is entered, the change is recorded and mail is sent to the appropriate people about this status change. Following this, control is returned to the author's menu. The new menu will look something like (assume "co" was entered as the new state):

```
                    Conference: cccc

     Author      Paper Status      Status Date
     ------      ------------      -----------
     janning     SUBMITTED         Thu Jun 11 07:35:59 1986
     stachowi    COMMITTEE REVIEW  Thu Jun 12 07:34:11 1986

     Enter author's name whose status will be changed ('q' to quit):
```

As before, the user can quit or make additional status changes. Entering a "q" will return the user to the main MARRS menu.

12. U - Option 'Update Audience Member List'

This option allows a user to add audience members to a conference. Only the major professor of an already initialized conference (see the "c" option) can add a audience members to a conference.

The total number of participants (authors, committee members, and audience members) that can be in one conference is limited to 30.

12.1 Selecting the Conference

After the "u" option is selected from the main menu, a search is done to see what conferences are available in which the user is the major professor of.

If no conferences exist for which the user is the major professor, "No conferences exist for you" will be displayed and the user is returned to the main MARRS menu.

If only one conference exists for which the user is the major professor of, the user will skip this part on "Selecting the Conference" and go directly to the "Adding Audience Members" section.

If, however, more than one conference exists for which the user is the major professor of, the following display is shown:

```
                    ADDING AUDIENCE MEMBER(S)

    Conferences available:

         aaaa          bbbb          cccc          dddd


    Enter a conference name ('q' to quit):
```

At this point a valid conference name, from the list given, should be entered. If a 'q' is entered, the user is returned to the main MARRS menu. If an invalid conference name is entered, the user is informed of the "Invalid conference name" and prompted for another conference name.

When a valid conference name is entered, the user is allowed to add new audience members.

12.2 Adding Audience Members

Following the selection of a conference, the next display shows:

```
     Adding new audience members to this conference

     Enter login id of audience member #4
      (Enter '.' when done):
```

The audience member's login name must be a valid login name on the system. If it is not, the user will be informed of this and will be prompted for another name. Note: In the above example, it is assumed that three audience members have previously been added.

An audience member's name cannot be the same as any other participant already entered into the conference. When a valid audience member's name is entered, the next audience member is prompted for.

When all audience members have been added, enter a '.' at the audience member's id prompt. At this point, appropriate directories are created and files updated. The user is then returned to the main MARRS menu.

Teleconferencing and the MARRS Computer Conferencing System

by

THOMAS JOSEPH STACHOWICZ

B. S. E. E., University of Toledo, 1980

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

An Abstract of a Master's Report

This paper presents MARRS (MAster's Report Reviewing System), a conferencing system, designed to provide a mechanism for tracking Master's Reports through the various stages of revision. The system is implemented on a UNIX based system. The roles of the participants in a MARRS conference are discussed in this paper.

MARRS is designed to be simple and easy to use by a conference of users. It uses existing UNIX utilities along with routines specially designed for this system. An individual MARRS conference is created for each master's project, which can consist of multiple papers. The individuals involved in a MARRS conference include a major professor, committee member(s), author(s), and an audience. A MARRS conference administrator maintains all conferences in the conferencing system. Each paper travel through various states during the review prior to publication.

A literature review, covering the various types of teleconferencing, is included in this paper. Audio, audiographics, video, and computer teleconferencing are discussed.