Designing and Implementing a Computer Conferencing System to Manage and Track Articles Through the Revision Process

by

Patricia Dock

B. A. University of West Florida, 1979

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

1984

Approved by:

Richard A. McBride
Major Professor

CONTENTS

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

LIST OF FIGURES

1. Introduction

1.1 Overview

This paper provides an overview of the services developed to manage and track articles for distribution to a community of users. The collection of these services has been named ARTHER (ARTicle HandlER). These services are simple and easy to use by the casual computer user. Machine dependent parameters such as logins and directories are hidden. The commands have a good human interface rather than one easily managed by a program. Most of the services may be customized to the preference of the individual user, others are customized to the preference of the owner of the subject.

The paper is organized in four chapters. The Introduction continues with a high level overview of ARTHER. Several terms are defined for the context of the paper. A high level description of the design assumption are presented, as well as an explanation of the possible applications for the service.

The second chapter covers the range of services and provides guidence for first time users. Also included are the types of input that are acceptable by ARTHER. Examples of the system in use are presented and explained.

The third chapter contains comparisons and contrasts between this system and several system which provide simular services. Included in these systems are VMSHARE, TELECENTER, and SCOOP.

The final chapter is a conclusion containing the problems and suggestions for future enhancements.

Four appendixes are attached. They contain manual pages for the user, manual pages for the administrator, a copy of the code to demonstrate an implementation for part of the project, and finally, a cross referencing of the functions and the variables.

A partial implementation was developed based on a subset of the described services. The implementation, written in C language,[1] and intended to run on a UNIX [2] operating system provides the services described for managing the articles.

An existing framework for electronic mail is assumed as well as a means of viewing files. The screen editor "vi" is utilized in the implemented version but could easily be replaced by any editor or screen viewing mechanism.

1.2 High Level Description

The purpose of ARTHER is to manage and track articles from the time of their submission for publication through the reviewing process until they are accepted for final publication.

---

1. C is a high level programming language that was developed by Bell Telephone Labs.

2. *UNIX is a Trademark of Bell Laboratories

In the context of this paper, several terms that normally have broad scopes will be applied to specific items. For further discussion, several of these terms are defined.

1.2.1 *Original Author* The phrase "original author" is intended to refer to the person whom is actually writing the article.

1.2.2 *Owner of the Subject* The phrase "owner of the subject" refers to the designated group coordinator. This is the person whom makes the overall decisions such as whom will review the article and whether an article is ready to be published.

1.2.3 *Members of the Subject* The term "members of the subject" refer to a list created by the ARTHER ADMINISTRATOR. All valid original authors and reviewers are subsets of this list.

1.2.4 *Reviewers* The phrase "reviewer" refers to a designated member of the group who has permission to review on the article prior to the acceptance for publication. This user has an opportunity to direct comments to the original author which are readable only by the original author and the owner of the group.

1.2.5 *Mail* The term "information is sent" implies that electronic mail containing this information is sent to the individual without human intervention.

1.2.6 *Informed* The term "is informed" implies that when the service is utilized, the individual will see titles for the appropriate articles with an indication of the type of action expected from them. This may happen at all logins, the first login

of the day, or only on demand. The choice is settable by the individual user. The first two choices never exclude the ability to perform the command on demand.

1.2.7 *Submitted* An article is submitted any time the original author decides to inquire if the owner of the subject considers the article acceptable for publication.

1.2.8 *Status of the Article* The status of the article is always at the discretion of the owner of the subject. The submission of the article is the stimulus to the original author to determine the status of the article. The STATUS can be ACCEPTED FOR PUBLICATION. REJECTED, or ACCEPTED FOR REVIEW.

1.2.9 *Accepted for Publication* The status "accepted for publication" implies that the article is considered in a final form and is ready to be viewed by the general public.

1.2.10 *Accepted for Review* The status "accepted for review" implies that the article is in a form such that a designated list (REVIEWERS) of experts should be asked to comment on the article.

1.2.11 *Rejected* The state "rejected" implies that the article is in the form such that the original author should make revisions and resubmit the article at a later date.

1.2.12 *More on the Status of the Article* The original author submits an article to the owner of the subject for review. The owner of the subject then determines the current status of the article. The article can be rejected, accepted for review, or

accepted for publication.

The current status of the article as well as accompanying comments are sent (i.e. mailed) to the original author. In the case of a rejection, this is the only apparent action taken.

If the article is accepted for review, the owner of the subject must designate a list of reviewers. The article is then available for review by this list of reviewers. Each of the reviewers have the opportunity to read and comment on the article.

Papers which have been either rejected or accepted for review can be re-submitted to the owner of the subject at the discretion of the original author.

Articles which are accepted for publication are available for reading by either a designated list or to the general public under a predetermined category.

1.2.13 *Comments* Comments consist of data which is entered via a command which contain opinions and suggestions concerning the article.

All comments are categorized as either public or private. A comment entered by a reviewer when the article is in the state of being reviewed (i.e. has been "accepted for review"), is considered private and are read and writable by the person entering the comment and the original author. The owner of the subject may or may not have access to the comments depending upon the option specified to the ARTHER ADMINISTRATOR when requesting the creation

of the subject. Their existence is acknowledged only by the tracking mechanisms which indicates whom commented on an article. The original author of the article is responsible for removing these comments.

Comments on articles which have been accepted for publication are considered public. These comments are readable by all, but are writable (thus removable) only by the owner of the subject. After three months the comments are automatically removed from the system.

1.3 Design Assumptions

When organizing the design assumptions, two categories were considered, system resources and user interfaces. In order to conserve the system resources, three things were done.

1. Only single copies of files are retained.
2. Minimal spin off of processes are utilized.
3. The installation of the code is totally automated.

For the user interface four things were done.

1. All commands have help functions.
2. All the commands have the same general features.
3. Users do not need to know or have access to information which is unnecessary to their function.

4. The user can modify the feature to their own tastes.

2. Description of ARTHER

2.1 Range of Services

The ARTicle HandlER services fall into several broad categories:

- User services
  - Customizing the services to one's own preferences
  - Creating articles
  - Reviewing articles
  - Tracking status of articles, or a user
  - Entering comments
  - Removing comments
- Administrative services for the owner of a subject
  - Creating, changing, removing specific lists
  - Designating status of the articles
  - Tracking status of the subject, articles, or members associated with a subject
- Administrative services for the ARTHER administrator

- Creating, changing, removing the subject

- Creating, changing, removing the lists of valid members

2.2 Getting Started

The ARTHER administrator must designate a directory under which all of the code and data for the system will exist. One's directories must be made [3] by the administrator. This directory must be named "src". The code should all be placed under the .../src directory. The administrator must be in the ".../src" directory when executing the "new_system" command. This command will make all the necessary changes and then execute the "make" file. It will build all the necessary directories and place the code in the ".../bin" directory.

On a continuing basis the ARTHER administrator must create subjects. Either the ARTHER ADMINISTRATOR creates a subject and designates an owner or a request is made to the ARTHER ADMINISTRATOR for a new subject with a specified owner. For each subject an owner must designated and a list must be generated which contains all valid members of a group associated with the subject. To be eligible to review a article one must be a member of this group.

The subject owner may create general lists for use in

---

3. Kernigan, Pike, p. 25.

designating lists of reviewers for individual articles.

For individual users of the system, several environment variables must be initialized when a user logs onto the system. These variables are used for various purposes such as setting program default options. The command PROFILER builds a default .ARTHER in the user's home directory. It then allows the user's to select their own values for command options. Finally it modifies the users .profile [4] in the HOME [5] directory to initialize .ARTHER and starts up the procedure at login.

The startup procedure provides information concerning the status of articles which have changed since your last login.

2.3 Tracking

For tracking purposes it may be useful to inquire into the current status of an article, an individual user, or a subject. Varied information is available for each status inquiry by a combination of options and permissions for the commands. When inquiring about an article, any user may view:

- the current status of an article
- the reviewers list

---

4. Kernigan, Pike, p. 36.

5. Kernigan, Pike, p. 36.

- public comments
- the names of those entering public comments

Additionally the owner of the subject and the original author may view:

- the dates of each status change
- a list of those who reviewed the article
- a list of those who submitted comments on the article
- a list by reviewers involved with the article
- a list of reviewers who have commented on the article.

Finally the original author may see the status and contents of the private comments for this article.

Inquiries could also be made concerning the status of an individual user. Both the author and the owner of the subject would be able to view a list of articles for which the user is the original author or the reviewer.

2.4 Types of Input

This section describes the different methods for interacting with ARTHER.

Each service is invoked by entering the name on the command line. Following the service name , one may enter keywords or other information such as the name of a user or article. Options and "namevalues" may be specified in any order,

although some services do assign special meaning to the order of the "namevalues".

After being invoked, the services prompt for any additional information that is required. There are three types of prompts: field, text input, and menu. Each is illustrated below.

At any time when utilizing either menu or field input one may type a question mark followed by a carriage return for more information and where applicable a list of acceptable responses.

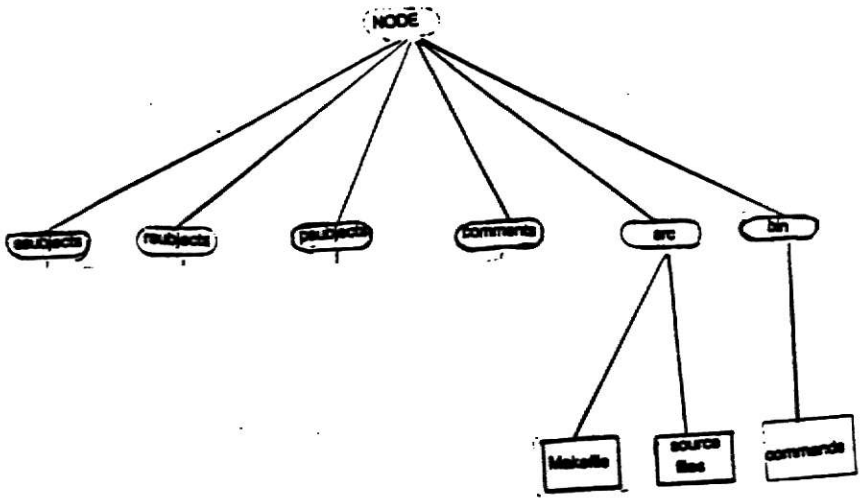
When using the text input mode, one terminates the text input by typing a period followed by a carriage return at the beginning of the line.

2.5 System in Use

To help understand ARTHER, examples of the system in use are presented. The bold print represents input by the user. After each example, a "pictorial" example of the state of the system is presented. The ovals represent directories and the boxes represent files.

Assuming the system is initialized and the executable modules are in ready, the system is used in this manner.

Figure 1. System Configuration - Initial state



The ARTHER administrator can "create" a subject. This is accomplished by executing the "artsubject" command. Figure 2 is a visual representation of the screen display when the "artsubject" command is executed.

Figure 2. Creating a Subject

```
artsubject
SUBJECT: test1
OWNER: dock

A list of valid members of the subject must be created.
All valid reviewers of articles in this subject must be
members of this group.


Enter names one at a time
when finished type a period
followed by a carriage return
For help type a question mark.
name: dock
name: smith
name: jones
name: .

Do you wish to edit the file or quit? q
```

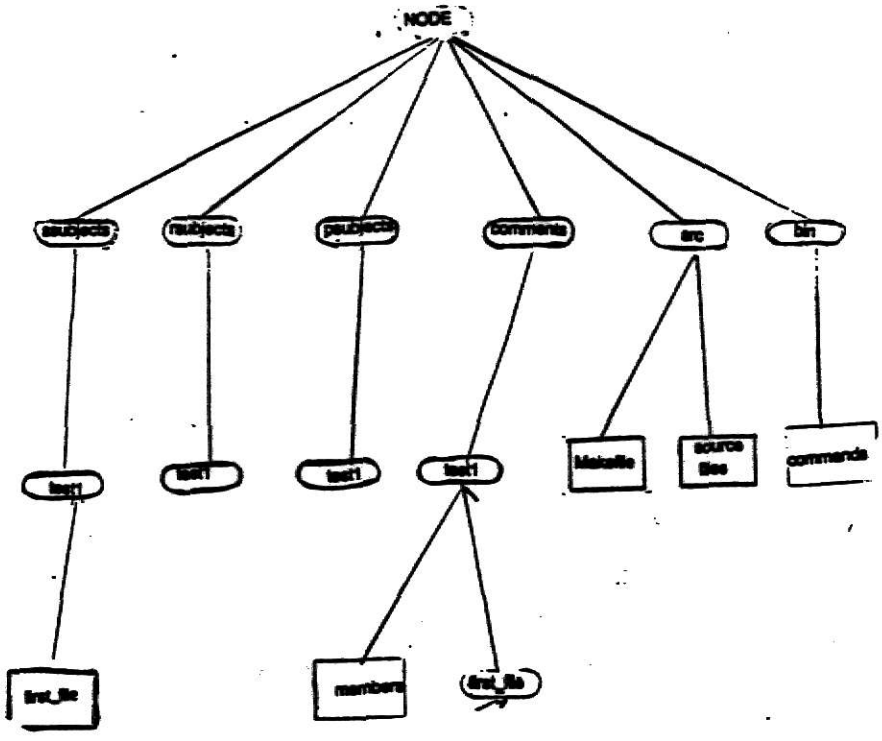
Dock, Smith, and Jones would all receive mail indicating that they should add "test1" to the ARTSUBJECTS in their .artprofile by means of the artprofile command. To continue to demonstrate the system services, let's assume Smith now has an article to submit to the owner of the subject. Remember Smith is a valid member of the subject "test1". The artsubmit command would be used. This command is displayed in Figure 3.

Figure 3. Submitting an Article

```
artsubmit
SUBJECT: test1
TITLE:  first file
Note: "first_file" will be used for name of the article.
File to be copied:   filetest1
```

The system state would have been altered again. Several directories would be "made". Figure 4 displays the current state of the system after the command "artsubmit" was completed.

Figure 4. System Configuration - artsubmit



Dock is the owner of the subject "test1". In our scenario let us assume Dock has ARTHER initialized to execute with each login. When Dock logs in, "art +n +s" is executed. At any time this command could be executed by hand. Figure 5 displays the output from this command.

Figure 5. Notification of Submitted Article

```
art +s  +n

"test1" subject for submission:
        first_file
```

Since a member of the subject has submitted an article for review, the system has informed the subject owner of the existence of the article. At the convenience of the subject owner the article may be reviewed. The "art +s +a" or "art +s" or "art +s test1" command are all means of performing this task. Figure 6 and 8 demonstrate one means of reviewing the article.

Figure 6. Viewing a Submitted Article

```
art +a  +s

    subject:   test1    article:  first_file

"vi" will be used to review the article.
You will be unable to write the file.
To enter comments for the author to read,
simply type ".:!artcomment" and you can enter
comments. REMEMBER: the author will know what
article you are commenting upon, but not where
you are in the program.  To exit the article
simply type ":q".
```

After the user types a carriage return, the article is viewed utilizing "vi" and the user may enter comments using the "artcomment" command. When viewing the article, if one were to type

```
:!artcomment
```

artcomment is executed. Figure 7 is an example of the execution of this command. An example of the artcomment example follows:

Figure 7. Entering Comments

```
artcomment
 Enter your comments a line at a time.
 When you have completed your comment,
 begin a line with a period (.)
 followed by carriage return.
This can be any text that the user wishes to
type. It will be placed in a file called comments.
```

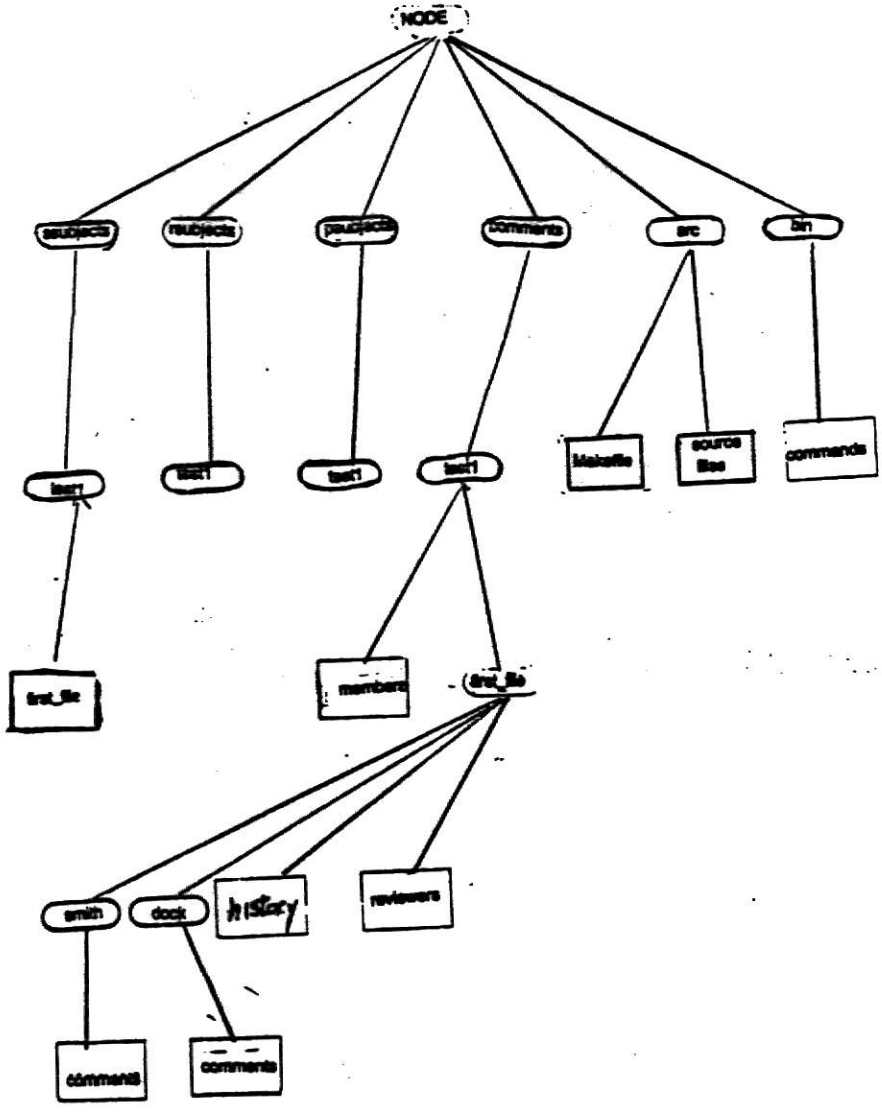
Once the reviewer is finished with the artcomment command and has exited vi, the command begun in Figure 6 is completed. The list of reviewers for which the owner of the subject is prompted must be valid members of the subject. If an invalid reviewer is entered, the command will inform the owner that an invalid user has been entered and not accept the entry.

Figure 8. Viewing a Submitted Article continued

```
Do you wish to accept for publication,
accept   for   review , or reject ?
publish, review, or reject : rev

Enter names one at a time
when finished, type a period
followed by a carriage return.
For help type a question mark.


name: dock
name: jones
name: ?
Enter names one at a time
when finished, type a period
followed by a carriage return.
name: .

Do you wish to edit the list or quit ? q


Do you wish to view the next file or quit?
next, quit ? [next]  n
```

Assuming that the prompts were answered as in the Figure 8, the new configuration follows. Several directories have been added to the system.

Figure 9. System Configuration - art +s



In our example, the status of the article in Figure 8 was determined to be "accepted for review". Several changes are in progress: Smith would receive mail indicating that the article was

accepted for review; Next time "art +r" was executed by Jones or Dock they would be notified that articles for review exist. This is illustrated in Figure 10.

Figure 10. Notification of Articles for Review

```
art +r  +n

"test1" subject for review:
        first_file
```

There are countless examples of commands that could be made, but the previous ones are sufficient to explain the different types of input.

3. Justification and Comparisons

3.1 Justification

The specification and resulting design of ARTHER emerged from an existing problem which the author has encountered many times in the span of her software career.

In a classroom situation, often group projects are a way of life. Several systematic means of organizing material can be devised by individual groups. However it is desirable to have an automated service existed which would not only organize a group's output, but also allow the instructor to monitor the progress of the group as well as assessing each individual's contribution to the project.

Take home exams could be entered on the computer and answered

on line. The instructor could grade them at own pace and return the grades in "comments" to the pupil.

In the Administrative Situations the Service could be utilized to help track and organize papers in progress. A very appropriate example would be the Master's Projects here at Kansas State University. A graduate advisor would own subject(s). The members of the subject(s) would be the graduate students whom are currently writing Master's Projects. As the Projects progress the Instructor would have a history of the project as well as easing the coordination of the papers and comments with professors.

Another situation in which ARTHER might prove an aid would be in attempting to reach a consensus concerning the content of a "core" course at a University. A description of changes could be available to all of the professors for opinion. Consequently, comments could be handled easily.

In a developmental environment, there is always living documentation (i.e. documentation which updated whenever a change is made to the project) associated with projects which eventually gets lost in the shuffle. Only recently have people become aware of how important it is to preserve the original requirements, specifications, and high level design for the project. These original documents all undergo many modifications before they are finally accepted. Thus copies of documentation are always suspect as to whether they are the most recent copy, or if in fact they are the true desired "original" copy. The reason for this is

that often one version exists while the another is being written or reviewed.

In general many areas where such a system such as ARTHER is needed exist. ARTHER solves the problem of keeping track of a system's documentation by separating the different users allowed to change and access individual documentation. The individual whom is the author of the documentation is the only person whom can change the document. The designated reviewers know that they are viewing the latest version of the document. The reviewers also know when ever a change has been made to the document. After the reviewers have had the chance to review and comment on the changes, the original author may again submit the article to the owner of the subject and the subject owner would have the option of accepting the article for publication. If accepted for publication, the public would have access to the article. If not yet acceptable for publication, the entire process begins again. But no confusion as to which version one is viewing. It is the newest version.

Understanding some of the possible applications for ARTHER, on can then begin to contrast ARTHER with some of the existing systems designed to perform the same functions. Following the specification, design and the majority of the implementation of ARTHER, a thorough search of existing systems was performed. Surprising in some ways and utterly expected in others, the existing systems resemble the design of ARTHER fairly closely. The areas where they differ seem to have direct trade offs in features.

When identifying systems with which to contrast ARTHER, the following definition for computer conferencing emerged.

> Conferences are a common writing space for group deliberations. Upon accessing a conference, users are brought up to date in the proceedings. Membership is controlled by a moderator. Participation is usually asynchronous but may be conducted in "real time". Conferences may be a few weeks to several years in duration, and the size may range from 2 to more than 50 members. Some conferences may be "public" or open all members of a given system. [6]

ARTHER certainly meets all the criteria put forth in this definition. It provides a common writing space for a group. The membership is controlled on two levels. the ARTHER ADMINISTRATOR controls the creation of groups while the subject owners control membership in the individual groups. No upper limit is placed on the number of members in individual groups. Some of the articles are "public" while others are restricted. With this definition in mind, several existing systems were identified for comparison with ARTHER.

3.2 Comparisons and Contrasts

3.2.1 *VMSHARE* VMSHARE is a computer conferencing facility developed in 1976 for use within Share, Inc,. an IBM users' group. It originated from the need to solve problems of continuity, planning, and communication between meetings. [7]

---

6. Kerr, p. 3.

7. Daney.

VMSHARE runs under CMS components of the IBM VM/370 operating system. It is written in the EXEC language (the standard command macro language of the 370.)

- As with ARTHER existing commands were utilized whenever possible.

- Unlike ARTHER, VMSHARE is written in an interpreted language. The trade off is ease of change for the interpreted language versus the speed of the compiled language.

- Unlike ARTHER, VMSHARE has restrictions on overlapping conference members.

- As in ARTHER, files are the basic elements of the system

- The owner of the file in VMSHARE creates the access list. In ARTHER the owner of the file (referred to as the author of the article) submits the article to the owner of the subject whom creates an access list (the reviewers list) or decides that the file could be a public file (accepts for publication.)

- Unlike ARTHER, VMSHARE is networked over a number of different machines. But all of these machines must be IBM 370s. See "Future Enhancements" for notes on networking.

3.2.2 *SCOOP* System for Computerization of Office Processes (SCOOP) is used to manage the editorial process for papers submitted to the Management Applications Section of Communications

of the ACM and for some papers submitted to the ACM Transactions on Database Systems. It provides a single interface for all office personnel to the specialized system. [8]

Prior to incorporation of SCOOP, the manual system was totally reactionary. When a stimulus occurred (say a paper was received in the mail) some resulting action usually took place. But the absence of some expected stimuli did not necessarily produce the desired action.

- Like ARTHER , SCOOP interacts with the existing electronic mail on the system.

- Since working with individuals spread over the country and not necessarily users who frequently log on, some sort of human intervention, such as conventional mail, is occasionally utilized in SCOOP.

- To submit a "newspaper article" to SCOOP, one executes a commands which prompt for author, the journal (corresponds to the subject in ARTHER), and the title of the newspaper article. To submit an article to ARTHER, one executes a command which prompts for author, the subject, and the title of the article.

- The article being submitted to SCOOP is the stimuli which

---

8. Zisman

results in a communication from the system with the journal editors asking for names of referees. The article being sub mitted to ARTHER is the stimuli to the subject owner to enter a list of reviewers.

- If no referee is suplied by the journal editor in a specified amount of time, another attempt is made to communicate with the editor. In ARTHER, every time the system is utilized, the subject owner is reminded that the article needs to be viewed. The owner is prompted for the list of reviewers when this command is executed,

- Potential referees in a SCOOP based system have an opportunity to refuse to act as referees for articles. In ARTHER this is not an option, but no attempt is made to force the reviewing either.

- In SCOOP as in ARTHER, comments are sent to the author of the article as well as the editor (subject owner in ARTHER). The owner of the subject in ARTHER and the editor in SCOOP make the decision as to whether to accept or reject the article.

- SCOOP was designed for "finished articles. ARTHER was designed to handle articles which are truly "living" documents. Consequently much history is available through ARTHER on an individual basis. When someone wishes to view the history of an article in SCOOP, one must scan all the history for all of the instances of all the articles in the journal for all of the information. This seems to be a limitation in the

implementation rather than in the design of the system.

3.2.3 *TELECENTER* Telecenter is a computerized conferencing system which exploits tools available on a UNIX based system. It was implemented with approximately one man weeks worth of effort.

- The design of Telecenter was meant to be a starting point for the users. It can be customized and enhanced by individual groups. It consists of a set of modifiable functions for use by groups in communicating through the computer.

- As in ARTHER, the reviewers can create conferences, enter new comments, and view own status in the conference. However it is different from ARTHER, in that the user can modify comments.

- In TELECENTER, when a reviewer submits a comment, the commenting process is accomplished through linking files in appropriate directories. It is not clear if the comment is identified with the person entering the comment or only with the conference.

- In TELECENTER, the users who have not viewed the article are linked to the article. When determining the number of users who have not reviewed the article, one may simply count the number of links to the file and you have a number of viewers still needing to view the article. In ARTHER, the names of the individuals as well as the number of individuals are viewed as the important information.

- TELECENTER is simular to ARTHER, when an article is submitted. In TELECENTER, a function is called which sets up a directory structure for the file corresponding to the article. In ARTHER, the artsubmit command creates a directory structure for the file in an existing directory structure which was created when the artsubject command was executed.

- Unlike ARTHER, TELECENTER makes no distinction between viewers who review the article and do not comment, and reviewers who comment on the file. Likewise in TELECENTER no information concerning "dates" is kept.

- Most of the functions which are utilized in TELECENTER are written in the UNIX system's standard command-interpreter language. This results in a trade off between adaptability by the individual users for their own needs versus efficiency. In ARTHER, most of the commands are written in "C" while a few, which are most likely to be modified by the individual user, are written in the interpreter language.

- Like ARTHER, the only networking implemented with TELECENTER is through the use of existing tools such as "cu".[9]

3.2.4 *SCCS* SCCS is an acronym for Source Code Control System. SCCS was designed to control changes to source code for storage, update, and retrieval of all versions of a module. It controls

---

9. Kernigan, Pike, p. 39.

updating privileges, identification of load modules by version and the record keeping of whom made which change, on which version the changes were made, and why the change was made.[10]

- Unlike ARTHER, SCCS is not a conferencing system. It is designed to handle large entities which will have many relatively small changes.

- Unlike ARTHER, it was designed for an environment in which one wishes to keep the "old" versions available. In ARTHER, one is attempting to rid oneself of the multiple copies of old articles.

- SCCS does keep records internal to the file as to whom made a change when, where, and why. This information is obtainable in several formats. But the intent is somewhat different from the intent of the records kept by ARTHER. The purpose of SCCS is to have the ability to recreate all versions of the module at any time. The intent of ARTHER is to have the latest version which is undergoing change and to have the ability to keep this evolving version separate from any existing version while controlling access.

- The access limitation in SCCS is through existing UNIX permission structure. This is limited to the owner, the group, and "all". In ARTHER the limitation is more precisely

---

10. Rochkind.

defined for the individual articles.

- SCCS is different from ARTHER, in that no recored keeping is done as to anyone viewing an article.

- In SCCS the concept of "comments" are simply "code" comments entered by the user whom is modifying the code.

- SCCS serves an important function but it is a completely different purpose from ARTHER.

4. Summary and Future Enhancements

ARTHER is a service which provides a tracking and handling mechanism for articles. The design is such that it is general enough to use in many different areas but specific enough to offer many features not included in other services. It is intended to have an appointed administrator who will perform some of the basic initializations of the groups but all of the procedures for set up are automated.

The design assumptions took into consideration "user friendly" niceties which make the system easy to use. Only a very basic interface to the tracking system is set up with this implementation. This is an area which could be expanded to encompass much more work.

The whole area of tracking and handling of articles is wide open. Security is a major issue both in an educational environment as well as in a developmental environment. This service has

attempted to answer these problems and make the user feel as comfortable as possible that the articles and the comments are accessible only to those with a need to know.

This implementation has not addressed the issue of networking. Certainly any UNIX system which has the command "cu" [12] has access to the system. If the UNIX command "uucp" [12] exists on multiple systems it would be possible to network the system. Several issues would need to be addressed. When would updating across systems take place? Would one build the intelligence into the system to decide whose information is sent to which machine login? That is, if one has a login on two machines, would both machines inform the user of changes, or would only one machine have the service active.

In general, this was an extensive project, but could be added to indefinitely. It is meant to be expanded to meet the needs of the users.

---

12. Kernigan, Pike, p. 39.

BIBLIOGRAPHY

[1] Daney,Charles,"The VMSHARE Computer Conferencing Facility", Computer Message System, Uhlig, R. P. (editor). North-Holland Publishing Company. IFIP,981.

[2] Grubb, Ralph E., "Student Control :Exploration in CAI", IBM Corporation,

[3] Kernighan, B.K.,Pike, Bob,"The UNIX Programming Environment" Printice-Hall, Inc., Englewood Cliffs, New Jersy.

[4] Kernighan, B.K.,Ritche,D.M.,"The C Programming Language", Printice-Hall, Inc., Englewood Cliffs, New Jersy.

[5] Kerr, E. B., Starr, R. H.,"Computer-Mediated Communication Systems Status and Evaluation",Human Communication Research Series, Academic Press, New York, New York, 1982.

[6] Pearson, M. M. L., Kulp, J. E., "Creating an Adaptive Computerized Conferencing System on UNIX". Computer Message System, Uhlig, R. P. (editor). North-Holland Publishing Company. IFIP,981.

[7] Rahmlow, H.F., Fratini, R.C., Ghesquiere,J.R.,"PLATO",The Instructional Design Library,Educational Technology Publications, Englewood Cliffs, New Jersy.

[8] Rochkind, Marc J., "The Source Code Control System" IEEE Transactions on Software Engineering, Volume SE1, No. 4, December, 1975.

[9] Wasserman,Anthony I., Information System Design Methodology", Tutorial on Software Design Techniques,Third Edition.

[10] Zisman, M. D., "Representation Specification and Automation of Office Procedures" Ph.D. dissertation, University of Pennsylvania, Pennsylvania, 1977.

Appendix 1

User Command Pages

NAME

art

SYNOPSIS

art [+[anocge]] [+[rs]] [<subjectname>] [<title>]

DESCRIPTION

Art is the means by which one views articles and titles of the articles which are currently being "managed" by the ARTHER system.

r: option indicates one wishes to view articles considered under review. Only valid reviewers can utilize this option.

s: option indicates that one wishes to view articles which have been submitted and for subjects which one is the valid group owner. The absence of s or r indicates that one is intersected in articles which are "published".

a: prints the actual articles which are in the appropriate environment.

o: prints the titles of articles which are in the appropriate environment.

g: prints name of the existing subjects

c: prints the number of articles which are newer than the last time the user printed out the articles.

e: prints name of all the existing subjects and the titles of the articles in all of the subjects.

n: prints the titles of articles which are newer than the last time

SEE ALSO

artsubmit artprofiler artsubjects

NAME

artsubject

SYNOPSIS

artsubject

DESCRIPTION

The artsubject command creates new subjects in the ARTHER service. The user is prompted for the name of the subject and the owner of the subject.

FILES

/ARTHERDIR/ssubjects directory is created.

/ARTHERDIR/rsubjects directory is created.

/ARTHERDIR/psubjects directory is created.

SEE ALSO

artsubmit art artprofiler

NAME

artsubmit

SYNOPSIS

artsubmit

DESCRIPTION

The artsubmit command accepts a file to be copied in to the ARTHER service for review by the owner of the subject. The user will be prompted for the subject, a title to be used for the article, and the file to be copied. The file to be copied can be a relative or a full path name. The subject must be a valid subject created with artsubject. The user must be a valid member of the subject.

SEE ALSO

artsubject art artprofiler

NAME

mklist

SYNOPSIS

mklist

DESCRIPTION

Mklist creates lists for use by the owner of the subject when creating list of reviewers for articles. Only the owner of the group or subject may create lists. The user will be prompted for the subject and a name for the list.

SEE ALSO

artsubmit

NAME

artprofiler

SYNOPSIS

artprofiler

DESCRIPTION

The artprofiler creates or changes the environment variables for the ARTHER service. One is prompted for a choice for the service to be run never, once a day, or every time one logs in. The user is also prompted for a chance to add subjects to their list of subjects to be considered for the environment.

FILES

/$HOME/.profile /$HOME/.artprofile

SEE ALSO

art artsubject artsubmit artprofiler

NAME

ml

SYNOPSIS

ml <filename> <filename>

DESCRIPTION

The ml command expects the first filename to contain a list of valid user ids on the system. The second filename should contain a "message" which will be mailed to every member of the list in the first filename.

SEE ALSO

mail art artsubject artsubmit artprofiler

NAME

arttrack

SYNOPSIS

arttrack

DESCRIPTION

The arttrack command is utilized to track the history of the ARTHER system. The user will be prompted for information as to which subject, reviewer, or article for s/he wishes to view. At any time one can enter a question mark for help concerning valid responses.

SEE ALSO

art artsubject artsubmit artprofiler

Appendix 2

Administrative Command Pages

NAME

new_system

SYNOPSIS

new_system

DESCRIPTION

The new_system command sets up the initialization necessary to implement the ARTHER service on a system. The system expects the user's current directory to be in a directory one level below the directory built especially for ARTHER. The recommendation is that a directory named /usr/ARTHER/src be created for this purpose. A copy of the source code should exist in this directory. The user must have write permission for the /usr/ARTHER directory. The command creates four directories and then performs a UNIX "make" of the system. It is not a problem if the user wishes to have a directory named other than "/usr/ARTHER", as long as the user's current directory is one level lower than the directory which is to be used as the ARTHER system directory.

SEE ALSO

make replace overwrite

NAME
replace

SYNOPSIS
replace

DESCRIPTION
The replace command replaces all occurences of one word with another.

SEE ALSO
overwrite arttrack

NAME
overwrite

SYNOPSIS
overwrite string1 string2 overwrite string1 string2 filenames

DESCRIPTION
The overwrite command first replaces all occurrences of string1 with string2 in files filenames. It then copies standard input to output after EOF.

SEE ALSO
replace arttrack

Appendix 3

Application Code

```
 1: #
 2: #define USE "art [+[agocen]] [+[rs]] [subname] [title]O
 3: #define ILL "   illegal option %s   usage:   %s"
 4: #define ILLTOG "   illegal option %s and %.1s
    . togetherOsage:   %s"
 5: #include "art.h"
 6:
 7:
 8: /* struct for command and env variable parsing */
 9: #define SAME 0
10: #define YES 1
11: #define NO 0
12: #define EMPTY ''
13:
14: char stdbuf[BUFSIZ];
15: char art_dir[BUFSIZE];
16: char art_subs[BUFSIZE];
17: char art_opts[BUFSIZE];
18: char art_excl[BUFSIZE];
19: char subnames[BUFSIZE];
20:
21:
22: jmp_buf save_addr;
23:
24:
25: int ncount;
26: long time();
27:
28:
29: extern char *strtok();
30:
31: main (argc,argv)
32: int  argc;
33: char *argv[];
34: {
35:         struct subjcts *subjcts[MAXBDS];
36:         int subjct_ct, substat, i;
37:         char sub[BUFSIZE];
38:         int print_item();
39:         char option;
40:         long time();
41:
42:         char types,*ptypes;
```

```
43:
44:         char opt[13];
45:         char typ[7];
46:         char *strchr(), *tempt;
47:
48:         char  *sdopt;
49:         int runcmd;
50:
51:         /*
52:         *   initialization
53:         */
54:
55:
56:         runcmd = YES;
57:         option = EMPTY;
58:         types = EMPTY;
59:         ptypes=&types;
60:         strcpy(opt,"agocenAGOCEN");
61:         strcpy(typ,"rsRS");
62:
63:
64:         i=1;
65:
66:
67:         while (argc -- >1)
68:             {
69:                 if(strncmp(argv[i],"+",1)!=0)
70:                 {
71:                     sprintf(subnames + strlen(subnames),"
                          %s",argv[i]);
72:                 }
73:                 else
74:                 {
75:                     if ((tempt=strchr(opt,argv[i][1])) == 0)

76:                     {
77:                         if ( (tempt=strchr(typ,argv[i][1]))
                              == 0)
78:                         {
79:                             printf(ILL,argv[i] , USE);
80:                             runcmd=NO;
81:                             continue;
82:                         }
```

```
 83:                         if ( types==EMPTY)
 84:                             strncpy(ptypes,tempt,1);
 85:                         else
 86:                         {
 87:                             printf(ILLTOG, ptypes, tempt,
                                   USE);
 88:                             runcmd=NO;
 89:                         }
 90:                     }
 91:                     else if ( option==EMPTY)
 92:                         strncpy(&option,tempt,1);
 93:                     else
 94:                     {
 95:                         printf(ILLTOG, ptypes, tempt, USE);
 96:                         runcmd=NO;
 97:                     }
 98:                 }
 99:             i++;
100:
101:         }
102:
103:         if (runcmd == NO) exit(1);
104:
105:         if ( types==EMPTY)
106:             strncpy(ptypes,"p",1);
107:         /*more initialization*/
108:         initialize (art_dir,art_subs,art_opts,art_excl,
               ptypes,option);
109:         get_subs (art_dir,art_subs,art_excl,subjcts,
               &subjct_ct);
110:         if ( subnames[0] == ' ' && option == EMPTY)
111:         {
112:                 for (i=0 ; i < subjct_ct ; i++)
113:                 {
114:                         if(subjcts[i]->excl == 1) continue;
115:                         outartcls(print_item,subjcts[i],
                               art_dir,NEW,ptypes);
116:                         subjcts[i]->stbuf.st_mtime = time
                               ((long *) 0);
117:                 }
118:                 update (subjcts,subjct_ct);
119:                 exit();
120:         }
```

```
121:
122:        /* process arguments */
123:        /* check for options, process command */
124:
125:
126:        if( option != EMPTY)
127:        {
128:
129:                if(subnames[0] == ' ')
130:                        process(subjcts,subjct_ct,art_dir,
                          "","",option,ptypes);
131:                else
132:                {
133:                        sdopt = strtok(subnames," ");
134:                        while(sdopt != NULL)
135:                        {
136:                                process(subjcts,subjct_ct,
                                  art_dir,sdopt,"",option,ptypes);
137:                                sdopt = strtok(0," ");
138:                        }
139:                }
140:        }
141:
142:        /* no option, just sub and/or news items */
143:        else
144:        {
145:                sdopt = strtok(subnames," ");
146:                substat = given_sub(sdopt,subjcts,subjct_ct)
                          ;
147:                if(GOOD) strcpy(sub,sdopt);
148:                else     sub[0] = ' ';
149:                sdopt = strtok(0," ");
150:                if(GOOD && sdopt == NULL)
151:                        process(subjcts,subjct_ct,art_dir,
                          sub,"",' ',ptypes);
152:                while(sdopt != NULL)
153:                {
154:                        process(subjcts,subjct_ct,art_dir,
                          sub,sdopt,' ',ptypes);
155:                        sdopt = strtok(0," ");
156:                }
157:        }
158: }
```

```
 1: /*
 2: convert arbitrary string to standard file name
 3: */
 4:
 5: #
 6: #include <stdio.h>
 7: #include <ctype.h>
 8:
 9: main(argc,argv)
10: int argc;
11: char **argv; {
12:
13:         int i,j,k;
14:
15:         k = 0;
16:
17:         for(i = 1; i < argc; i++)
18:         {
19:
20:                 j = 0;
21:                 while( argv[i][j] != ' ')
22:                 {
23:
24:                         if(isalnum(argv[i][j]) ||
25:                            argv[i][j] == '+' ||
26:                            argv[i][j] == '_' ||
27:                            argv[i][j] == '-' ||
28:                            argv[i][j] == '.' )
29:                         {
30:
31:                                 putc(argv[i][j],stdout);
32:                                 if(++k == 14) exit(0);
33:                         }
34:                         j++;
35:                 }
36:                 if(++k == 14) break;
37:                 if(i+1 == argc) break;
38:                 putc('_',stdout);
39:         }
40:
41:         putc('0,stdout);
42: }
```

```
 1:
 2: # include "art.h"
 3:
 4: cominfo(dirnm,nm)
 5: char *dirnm, *nm;
 6: {
 7:         char *getlogin(), s[BUFSIZE], sys[BUFSIZE];
 8:
 9:
10:         strcpy(nm,getlogin());
11:         sprintf(s,"%s/%s",dirnm,nm);
12:         chdir(s);
13:         sprintf(sys,"/usr/we/dock/artrecord
              reviewed article for publication
14:         system(sys);
15:         printf("
              the article.");
16:         printf("   You will be unable to write the
                file.");
17:         printf("   To enter comments for the author
              to read,");
18:         printf("   simply type
              can   enter");
19:         printf("   comments.  REMEMBER: the author will
              know what ");
20:         printf("   article you are commenting upon, but
              not where");
21:         printf("    you are in the program. . To exit the
              article");
22:         printf("   simply type
23:         printf("   Type a carriage return when ready
              for the file.");
24:         gets(s);
25: }
```

```
 1: /*
 2:      artcomment.c
 3:
 4:          Assumes that one is in the directory where
 5:          one wishes the comments to be written in a
 6:          file named comment.
 7:          If called from artI+s then the directory
 8:          should be  artdir/comments/$SUBJECT/$TITLE/$REVIEWER
 9:          for the articles which are being reviewed.
10:          The directory should be
               artdir/comments/$SUBJECT/$TITLE
11:          for files which are considered "published."
12:
13: */
14: #include <stdio.h>
15:
16: main()
17: {
18:          char s[90];
19:          char NEWLINE;
20:          FILE *fopen(), *fp;
21:          s[0]=' ';
22:          fp=fopen("comment","a");
23:
24:          printf( " Enter your comments a line at a time.");
25:          printf( " When you have completed your comment,");
26:          printf( " begin  a  line  with  a  period   (.)");
27:          printf( " followed by cariage return.");
28:          printf( " ");
29:          for (gets(s); s[0] != '.' && s[1] != ' ';)
30:          {
31:                  fputs(s,fp);
32:                  fputs('0,fp);
33:                  gets(s);
34:          }
35:          fclose(fp);
36: }
37:
```

```
 1:
 2:
 3: /*      delete.c
 4:  *      removes a file named fname.
 5:  *      called by delete(fname)
 6:  */
 7:
 8: #include "art.h"
 9:
10: delete(fname)
11: char *fname;
12: {
13:         if (unlink(fname)!=SUCCESS)
14:                 printf(" unable to remove list.");
15: }
16:
```

```
 1:
 2: /*      edits a list, giving an oportuinity to
 3:  *      keep or remove each entry in the list.
 4:  *      called edit(fname)
 5:  */
 6:
 7: #include "art.h"
 8:
 9: edit(fname)
10: char *fname;
11: {
12:         char tmpfile[BUFSIZE];
13:         char ans[10];
14:         char s[BUFSIZE];
15:         FILE  *fopen(), *tmp, *fp;
16:         int getpid();
17:
18:
19:         sprintf(tmpfile, "tmp%d", getpid());
20:         if((tmp=fopen(tmpfile,"w"))== NULL)
21:                 printf("Onable to open %s",tmpfile);
22:         if((fp=fopen(fname,"r"))== NULL)
23:                 printf("Onable to open %s",fname);
24:
25:
26:         while(fgets(s, sizeof s,fp)!=NULL)
27:         {
28:                 printf("%s: keep or delete [keep]?", s);
29:                 gets(ans);
30:                 switch(ans[0])
31:                 {
32:                 case '?':       printf (" keep:.  entry
                      remains in  list");
33:                                 printf (" delete: removes
                                      entry from list");
34:                                 printf("%s: keep or
                                      delete [keep]?", s);
35:                                 gets(ans);
36:
37:                 case 'k':       fputs(s,tmp);
38:                                 fputs("0,tmp);
39:                 }
40:         }
```

```
41:         fclose(fp);
42:         fclose(tmp);
43:         sprintf(s,"mv %s %s",tmpfile, fname);
44:         system(s);
45:
46: }
47:
48:
49:
50:
51:
```

```
 1: /*
 2:  *   get_subs
 3:  * read names and last modified times of subs
 4:  *
 5:  *      arguments
 6:  *              allsubs and     ptrs of allocated memory
 7:  *              usersub            where files are read in
 8:  *              allsub_ct and   number of subs found
 9:  *              usersub_ct         in each file
10:  *
11:  *      enough memory is allocated for usersub to fit all
       possible subs
12:  */
13:
14: #
15: #include "art.h"
16:
17: get_subs (art_dir,art_subs,art_excl,subjcts,subjct_ct)
18: char art_dir[],art_subs[],art_excl[];
19: struct subjcts *subjcts[MAXBDS];
20: int *subjct_ct;
21: {
22:         struct usersub usersub;
23:         struct stat stbuf;
24:         struct direct dirbuf;
25:         int i,fd;
26:         char *malloc();
27:         FILE *fopen(), *infile;
28:         char *env, *strtok();
29:         char fullname[BUFSIZE];
30:
31:         /* get sub names */
32:
33:         if(stat(art_dir,&stbuf)  == -1 ||
34:            (stbuf.st_mode & S_IFMT) != S_IFDIR)
35:         {
36:                 fprintf(stderr,"art:
                          ARTSUBSO,art_dir);
37:                 exit(1);
38:         }
39:
40:         if((fd = open(art_dir,0)) == -1)
41:         {
```

```
42:                 fprintf(stderr,"art: cannot read
                      sub0, art_dir);
43:                 exit(1);
44:         }
45:
46:         /* read in file names */
47:         *subjct_ct = 0;
48:         while(read(fd,(char *)&dirbuf,sizeof(dirbuf))>0)
49:         {
50:                 if(dirbuf.d_inc == 0) continue;
51:                 if(strcmp(dirbuf.d_name,".") == 0 ||
52:                    strcmp(dirbuf.d_name,"..") == 0)
                      continue;
53:                 /* check for directory, modified times */
54:                 sprintf(fullname,"%s/%s",art_dir,
                      dirbuf.d_name);
55:                 if(stat(fullname,&stbuf) == -1) continue;
56:                 if((stbuf.st_mode & S_IFMT) != S_IFDIR)
                      continue;
57:                 /* add information to subjcts */
58:                 subjcts[*subjct_ct] = malloc( sizeof
                      (struct subjcts) );
59:                 stat( fullname, & (subjcts[*subjct_ct]
                      ->stbuf));
60:                 if(art_subs[0] == ' ')
61:                       subjcts[*subjct_ct]->excl = 0;
62:                 else subjcts[*subjct_ct]->excl = 1;
63:                 strcpy(subjcts[*subjct_ct]->name,
                      dirbuf.d_name);
64:                 subjcts[(*subjct_ct)++]->new = 1;
65:                 }
66:
67:
68:         /* check for new subs */
69:         for(i = 0; i < *subjct_ct; i++)
70:         {
71:                 if(subjcts[i]->new == 0) continue;
72:                 subjcts[i]->stbuf.st_mtime = 0L;
73:         }
74:
75:         /* check for valid subs */
76:         env = strtok(art_subs,":");
77:         if (env[0] == '+') env++;       /* skip leading + */
```

```
78:         while(env != NULL)
79:         {
80:                 for(i = 0; i < *subjct_ct; i++)
81:                 {
82:                         if(strcmp(subjcts[i]->name,env) ==
                              0)
83:                         {
84:                                 subjcts[i]->excl = 0;
85:                                 break;
86:                         }
87:                 }
88:                 env = strtok(0,":");
89:                 if (env[0] == '+') env++;       /* skip
                      leading + */
90:         }
91:
92:         /* check for excluded subs */
93:         env = strtok(art_subs,":");
94:         if (env[0] == '-') env++;       /* skip leading - */
95:         while(env != NULL)
96:         {
97:                 for(i = 0; i < *subjct_ct; i++)
98:                 {
99:                         if(strcmp(subjcts[i]->name,env) ==
                              0)
100:                        {
101:                                subjcts[i]->excl = 1;
102:                                break;
103:                        }
104:                }
105:                env = strtok(0,":");
106:                if (env[0] == '-') env++;       /* skip
                      leading - */
107:        }
108:
109: }
```

```
 1:
 2:
 3: /*
 4:  *   given_sub
 5:  *                        -1   if a nonexistant sub is passed
 6:  *                         0   if NULL is passed
 7:  *                         1   if sub is a member of sublist
 8:  *                             and i = its index into the list
 9:  */
10:
11: #
12: #include "art.h"
13:
14: given_sub (sub,subjcts,subjct_ct)
15: struct subjcts *subjcts[MAXBDS];
16: int subjct_ct;
17: char *sub;
18: {
19:         int i;
20:
21:         if ( sub == NULL  || *sub == ' ')
22:                 return (-1);
23:      /* linear search thru sublist for sub */
24:         for(i = 0; i < subjct_ct; i++)
25:         {
26:
27:                 if(strcmp(subjcts[i]->name,sub) == 0)
                      return(i);
28:         }
29:
30:         return (subjct_ct);
31: }
```

```
 1: #define ARTDIR "/usr/we/dock"
 2: #include "art.h"
 3: #include <stdio.h>
 4: #include <signal.h>
 5:
 6:
 7: initialize(art_dir,art_subs,art_opts,art_excl,types,option)
 8: char art_dir[],art_subs[],art_opts[],art_excl[],*types,
      option;
 9: {
10:         extern _exit();
11:         char *env,*getenv();
12:         if (signal (SIGQUIT, SIG_IGN) != SIG_IGN)
13:                 signal (SIGQUIT, _exit);
14:
15:         /* get environment variables */
16:         strcpy(art_dir,ARTDIR);
17:         strcat(art_dir,"/");
18:
19:         if( (strcmp(types,"p")==SUCCESS) && (option=='e') )
20:                 strcat(art_dir,"comments");
21:         else
22:         {
23:                 strcat(art_dir,types);
24:                 strcat(art_dir,"subjects");
25:         }
26:         env = getenv("ARTSUBS");
27:         if(env == NULL || strlen(env) == 0)
28:         {
29:                 art_subs[0] = ' ';
30:         }
31:         else strcpy(art_subs,env);
32:
33: }
```

```
 1:
 2: /*      creates a list of name fname.
 3:  *      looks in file fcheck for validation of names
 4:  *      called creat(fname,fcheck)
 5:  */
 6:
 7: #include "art.h"
 8:
 9: listcreate(fname,fcheck)
10: char *fname, *fcheck;
11: {
12:         FILE *fopen(), *fp;
13:         char s[BUFSIZE];
14:         char sys[BUFSIZE];
15:         int answer;
16:
17:         if ((fp=fopen(fname,"a"))== NULL)
18:                 printf(" can't open fname");
19:
20:         printf(" Enter names one at a time");
21:         printf(" when finished, type a period");
22:         printf(" followed by a carriage return.");
23:         printf(" For help type a question mark.");
24:
25:         printf(" name: ");
26:         gets(s);
27:
28:         while (answer != TRUE)
29:                 switch (s[0])
30:                 {
31:                         case '?':
32:                                 printf(" Enter  names
                                      one  at  a  time");
33:                                 printf(" when finished,
                                      type  a  period");
34:                                 printf(" followed by a
                                      carriage return.");
35:                                 printf(" name: ");
36:                                 gets(s);
37:                                 break;
38:                         case '.':
39:                                 if (s[1] == NULL)
40:                                         answer = TRUE;
```

```
41:                                     break;
42:                             default:
43:                                     sprintf(sys,
                                          "/usr/we/dock/valid %s %s",s,fcheck);
44:                                     if (system(sys) == SUCCESS)
45:                                     {
46:                                             fputs(s,fp);
47:                                             fputs("0,fp);
48:                                     }
49:                                     else
50:                                             printf(" invalid
                                                  entry");
51:                                     printf(" name: ");
52:                                     gets(s);
53:                     }
54: fclose(fp);
55: }
56:
57:
58:
```

```
1:
2: mail(letter,mailinglist)
3: char *letter, *mailinglist;
4: {
5: char com[256];
6: sprintf(com,"/usr/we/dock/ml %s %s",mailinglist,
     letter);
7: system(com);
8: }
```

```
#include "art.h"

mklist(fname,dirname,fcheck)
char *dirname;
char *fname;
char *fcheck;
{
        int answer;
        char s[BUFSIZE];
        char a[BUFSIZE];
        struct stat statinfo;

        /* If the fname does not exist, give the user a
          chance to
           utilize an existing list.
        */

        if((stat(fname,&statinfo)==FALSE) && (dirname ==
          ' '))
        {
                for (answer=0; answer!=TRUE;)
                {
                        printf(" The list you have
                          indicated does not");
                        printf("exist, Do you wish to
                          duplicate an existing");
                        printf("list for this suartject ? ")
                          ;
                        gets(s);
                        switch ( s[0])
                        {
                        case 'y':       printf(" list: ");
                                        gets(s);
                                        sprintf(a,"%s/%s",
                                          dirname,s);
                                        if(stat(a,&statinfo)
                                          ==FALSE)
                                                break;
                                        answer=TRUE;
                                        sprintf(s,"cp %s
                                          %s",a,fname);
                                        system(s);
                        case 'n':       answer=TRUE;
```

```
36:                         case '?':       break;
37:                         }
38:                 }
39:         }
40:
41:         /*  If we are here, we have already copied the file
42:             if one was wanted.  If fname  does  not  exists,
43:             create it empty and prompt for names to add to
44:             it
45:         */
46:         if(stat(fname,&statinfo)==FALSE)
47:         {
48:                 listcreate(fname,fcheck);
49:         }
50:
51:         /* Give user chance to edit the file
52:         */
53:
54:
55:         printf(" Do you wish to edit the file or quit?");
56:         gets(s);
57:         if (strncmp(s,"q", 1)== NULL) return;
58:         edit(fname);
59: }
```

```
 1: #include "art.h"
 2: #define ARTDIR "/usr/we/dock"
 3:
 4:
 5: main()
 6: {
 7: int whoami;
 8: int answer;
 9: char dirname[BUFSIZE];
10: char fname[BUFSIZE];
11: char s[BUFSIZE];
12: char sy[BUFSIZE];
13: struct stat statinfo;
14:
15: /* Get the uid of the person executing command*/
16: whoami=geteuid();
17:
18:
19: printf(" Lists are created on a subect basis by the owner
      of the group.");
20: for (answer=0;answer!=TRUE;)
21: {
22:         printf(" subject: ");
23:
24:         s[0] = ' ';
25:         if (gets(s) == NULL) exit(1);
26:         switch (s[0])
27:         {
28:         case '?':
29:                 printf("\nter the name of the subject for
                      which you wish to");
30:                 printf(" create the list. You must be the
                      owner of the group");
31:                 printf(" Type quit to exit.");
32:         case NULL:
33:                 break;
34:         default:
35:                 sprintf(dirname,"%s/comments/%s",ARTDIR,s);
36:                 if(stat(dirname,&statinfo)==FALSE)
37:                 {
38:                         printf(" %s is not a valid
                              subject",s);
39:                         break;
```

```
40:                     }
41:                     if (statinfo.st_uid != whoami)
42:                     {
43:                             answer=TRUE;
44:                     }
45:                     else
46:                             printf(" You are not the owner of
                                  the group %s",s);
47:             }
48: }
49:
50:             printf(" Do you wish to create a new list, delete
                  an existing list,");
51:             printf(" or edit an existing list?");
52:             printf(" create, delete, or edit? ");
53:             gets(s);
54:             switch (s[0])
55:             {
56:             case 'd':
57:                     printf(" name of existing list: ");
58:                     gets(s);
59:                     sprintf(sy,"rm -f %s/%s",dirname,s);
60:                     system(sy);
61:
62:                     exit(0);
63:             default:
64:                     printf(" Name of the list you wish to
                          create or edit? ");
65:                     gets(s);
66:                     sprintf(fname,"%s/%s",dirname,s);
67:                     mklist(fname,dirname, "/etc/passwd");
68:             }
69: }
```

```
 1: #include "art.h"
 2:
 3:
 4:
 5: main(argc,argv)
 6: int argc;
 7: char *argv[];
 8:
 9: {
10: int answer;
11: char s[BUFSIZE];
12:         listcreate(argv[1], "/etc/passwd");
13:         for (answer=0;answer!=TRUE;)
14:         {       printf("Do you wish to edit the file or
              quit? ");
15:                 gets(s);
16:                 switch (s[0])
17:                 {
18:                 case 'e':
19:                                 edit(argv[1]);
20:                                 answer=TRUE;
21:                                 break;
22:                 case 'q':   exit(1);
23:                 case '?':   printf("Odit: examine the
                      entries one at a time");
24:                                 printf("Ouit: leave the list
                                      as it is.");
25:                                 edit(argv[1]);
26:                 }
27:         }
28: }
29:
```

```
 1: /*
 2:  *   newcount
 3:  *       counts number of artcl
 4:  */
 5: newcount (s,sub)
 6:          char *s,
 7:               sub[];
 8: {
 9:          extern int ncount;
10:          if(s) ncount++;
11:
```

```
1: #
2: #include "art.h"
3:
4:
5: onintr()
6: {
7:         fprintf(stderr,"0interrupt0);
8:         longjmp(save_addr, 1);
9: }
```

```
1: output(filename,outs)
2: char filename[],outs[];
3: {
4:         char t[256];
5:
6:         sprintf(t,"/usr/sccsbin/vi %s %s",filename,outs);
7:         system(t);
8: }
```

```
 1: #
 2: #include "art.h"
 3:
 4:
 5:
 6: print_item (f,sub,art_dir,ptypes)
 7: char *f, sub[], art_dir[],*ptypes;
 8: {
 9:         FILE *fd;
10:         extern int interactive;
11:         char s[BUFSIZE];
12:         int  n, i;
13:         char fname[BUFSIZE];
14:         char newfname[BUFSIZE];
15:         char nm[10];
16:         char comdir[BUFSIZE];
17:         static int firstitem = 1;
18:         int onintr();
19:
20:         if (f == NULL) {
21:                 return;
22:         }
23:         n=strlen(art_dir) - 9;  /*size of string ssubjects*/
24:         strncpy(comdir,art_dir,n);
25:         sprintf(fname,"%s/%s/%s",art_dir,sub,f);
26:         sprintf(newfname,"%scomments/%s/%s",comdir,sub,f);
27: start:
28:         if ((fd = fopen (fname, "r")) != NULL)
29:         {
30:                 register int c, ip, op;
31:                 struct stat sbuf;
32:                 char *ctime();
33:                 extern int ncount;
34:
35:                 ncount++;
36:
37:                 fstat (fileno (fd), &sbuf);
38:                 if (firstitem) {
39:                         firstitem = 0;
40:                         putchar ('0');
41:                 }
42:                 if (setjmp(save_addr))
43:                         goto finish;
```

```
44:                 if (signal(SIGINT, SIG_IGN) != SIG_IGN)
45:                         signal(SIGINT, onintr);
46:                 s[0] = ' ';
47:                 printf("   subject: %s        article:
                      %s\0,sub,f);
48:                 cominfo(newfname,nm);
49:                 output(fname,&s[0]);
50: finish:
51:                 putchar ('\0');
52:                 fclose (fd);
53:                 if (signal(SIGINT, SIG_IGN) != SIG_IGN)
54:                         signal(SIGINT, SIG_DFL);
55: prompt:
56:                 s[0] = ' ';
57:                 fprintf(stderr,"review, next, quit ? [next]
                      ");
58:                 gets(s);
59:                 switch (s[0])
60:                 {
61:                 case 'r':
62:                                 putc('\0',stdout);
63:                                 goto start;
64:                                 break;
65:                 case 'n':
66:                 case 'q':       if (strncmp(ptypes,"s",1)==
                      0) submit(fname,n,sub,f,comdir,nm,newfname);
67:                                 exit(1);
68:                                 break;
69:
70:                 case '?':       fprintf(stderr,"
                      %-10s%s\0,"review","print artcle again");
71:                                 fprintf(stderr,"
                                      %-10s%s\0,"next","go to next artcle");
72:                                 fprintf(stderr,"
                                      %-10s%s\0,"quit","terminate program");
73:                                 goto prompt;
74:                                 break;
75:
76:                 case ' ':       break;
77:
78:                 default:        fprintf(stderr,"unknown
                      command, please try again.\0);
79:                                 goto prompt;
```

```
80:                                     break;
81:                 }
82:                 putc('0,stdout);
83:         }
84: }
```

```
 1: #
 2: #include "art.h"
 3:
 4: process(subjcts,subjct_ct,art_dir,sub,item,option,types)
 5: struct subjcts *subjcts[MAXBDS];
 6: int subjct_ct;
 7: char *sub,*item,option,*types;
 8: {
 9:         int newsstat, substat,i,j;
10:         long time();
11:         int  print_item(), report(), count(), newcount();
12:         int flag;
13:
14:
15:         /* check option for all subs */
16:         if( isupper(option) )
17:         {
18:                 flag = 2;
19:         }
20:         else flag = 1;
21:
22:         switch (option )
23:         {
24:         case 'n':        /* report titles of new artcl */
25:         case 'N':
26:                 ncount = 0;
27:                 substat = given_sub(sub,subjcts,subjct_ct);
28:                 if ( GOOD )
29:                 {
30:                         outartcls(report,subjcts[substat],
                              art_dir,NEW,types);
31:                         if(ncount == 0) fprintf(stdout,
                              "There are no new artcl on
32:                         }
33:                 else if ( NOTHERE )
34:                 {
35:                         for (i=0 ; i < subjct_ct ; i++)
36:                         {
37:                                 if(subjcts[i]->excl == flag)
                                       continue;
38:                                 outartcls(report,subjcts[i],
                                      art_dir,NEW,types);
39:                         }
```

```
40:                 }
41:                 else if ( BAD )
42:                 {
43:                         fprintf(stderr,"
                              existing sub.0,sub);
44:                 }
45:
46:                 break;
47:
48:         case 'o':        /* report appropriate  titles of */
49:                         /* articles regardless of currency
                              */
50:         case 'O':
51:                  ncount = 0;
52:                 substat = given_sub(sub,subjcts,subjct_ct);
53:                 if ( GOOD )
54:                 {
55:                         outartcls(report.subjcts[substat].
                              art_dir,ALL,types);
56:                         if(ncount == 0) fprintf(stdout,
                              "There are no artcl on
57:                         }
58:                 else if ( NOTHERE )
59:                 {
60:                         for (i=0 ; i < subjct_ct ; i++)
61:                         {
62:                                 if(subjcts[i]->excl == flag)
                                       continue;
63:                                 outartcls(report,subjcts[i],
                                      art_dir,ALL,types);
64:                         }
65:                 }
66:                 else if ( BAD )
67:                 {
68:                         fprintf(stderr,"
                              existing sub.0,sub);
69:                 }
70:
71:                 break;
72:
73:
74:         case 'e':       /* report all titles of articles */
75:         case 'E':
```

```
 76:                  ncount = 0;
 77:                 substat = given_sub(sub,subjcts,subjct_ct);
 78:                 if ( GOOD )
 79:                 {
 80:                         outartcls(report,subjcts[substat],
                               art_dir,EVERYONE,types);
 81:                         if(ncount == 0) fprintf(stdout,
                               "There are no artcl on
 82:                         }
 83:                 else if ( NOTHERE )
 84:                 {
 85:                         for (i=0 ; i < subjct_ct ; i++)
 86:                         {
 87:                                 outartcls(report,subjcts[i],
                                       art_dir,EVERYONE,types);
 88:                         }
 89:                 }
 90:                 else if ( BAD )
 91:                 {
 92:                         fprintf(stderr,"
                               existing sub:0,sub);
 93:                 }
 94:
 95:                 break;
 96:
 97:
 98:         case 'c':        /* count new artcls */
 99:         case 'C':
100:                 ncount = 0;
101:                 substat = given_sub(sub,subjcts,subjct_ct);
102:                 if ( GOOD )
103:                 {
104:                         outartcls(newcount,subjcts[substat],
                               art_dir,NEW,types);
105:                         }
106:                 else if ( NOTHERE )
107:                 {
108:                         for (i=0 ; i < subjct_ct ; i++)
109:                         {
110:                                 if(subjcts[i]->excl == flag)
                                        continue;
111:                                 outartcls(newcount,
                                       subjcts[i],art_dir,NEW,types);
```

```
112:                        }
113:                }
114:                else if ( BAD )
115:                {
116:                        fprintf(stderr,"
                              existing sub.C.sub);
117:                }
118:
119:                if(ncount > 0) fprintf(stdout,"%d0,ncount)
                      :
120:                break;
121:
122:
123:        case 'a':   /* print contents of all artcls
              regardless of currency */
124:        case 'A':
125:                ncount = 0;
126:                substat = given_sub(sub.subjcts.subjct_ct);
127:                if ( GOOD )
128:                {
129:                        outartcls(print_item,
                              subjcts[substat].art_dir.ALL.types);
130:                        subjcts[substat]->stbuf.st_mtime =
                              time ((long *) 0);
131:                        update (subjcts,subjct_ct);
132:                        if(ncount == 0) fprintf(stdout,
                              "There are no artcl on
133:                        }
134:                else if ( NOTHERE )
135:                {
136:                        for (i=0 ; i < subjct_ct ; i++)
137:                        {
138:                                if(subjcts[i]->excl == flag)
                                      continue;
139:                                outartcls(print_item,
                                      subjcts[i].art_dir.ALL.types);
140:                                subjcts[i]->stbuf.st_mtime
                                      = time ((long *) 0);
141:                        }
142:                        update (subjcts,subjct_ct);
143:                }
144:                else if ( BAD )
145:                {
```

```
146:                            fprintf(stderr,"
                                  existing sub.O,sub);
147:                    }
148:
149:                    break;
150:
151:            case 'g':   /* report all sub names */
152:            case 'G':
153:                    substat = given_sub(sub,subjcts,subjct_ct);
154:                    if ( GOOD )
155:                            fprintf(stdout,"
                                  sub);
156:                    if (( NOTHERE ) && (strncmp(types,"p",1)
                          !=SUCCESS))
157:                    {
158:                            fprintf(stdout,"existing subjects
                                  in your environment:");
159:                            j = 0;
160:                            for (i=0 ; i < subjct_ct ; i++)
161:                            {
162:                                    if(subjcts[i]->excl == flag)
                                          continue;
163:                                    if((j++)%5 == 0) printf("
                                          ");
164:                                    fprintf(stdout,"%-15.14s",
                                          subjcts[i]->name);
165:                            }
166:                            putc('O,stdout);
167:                    }
168:                    if (( NOTHERE ) && (strncmp(types,"p",1)
                          ==SUCCESS))
169:                    {
170:                            fprintf(stdout,"existing subjects
                                  :");
171:                            j = 0;
172:                            for (i=0 ; i < subjct_ct ; i++)
173:                            {
174:                                    if((j++)%5 == 0) printf("
                                          ");
175:                                    fprintf(stdout,"%-15.14s",
                                          subjcts[i]->name);
176:                            }
177:                            putc('O,stdout);
```

```
178:                    }
179:                    if ( BAD )
180:                            fprintf(stdout,"
                          0,sub);
181:                    break;
182:
183:            case ' ':
184:                    substat = given_sub(sub,subjcts,subjct_ct);
185:                    newsstat = given_sub(item,subjcts,subjct_ct)
                          ;
186:                    ncount = 0;
187:
188:                    /* if a sub, print artcl there */
189:                    if(GOOD && newsstat == -1)
190:                    {
191:                            outartcls(print_item,
                              subjcts[substat],art_dir,NEW,types);
192:                            subjcts[substat]->stbuf.st_mtime =
                              time ((long *) 0);
193:                            update (subjcts,subjct_ct);
194:                            if(ncount == 0) printf("There are
                              no new artcl on
195:                    }
196:                    /* if sub & title, print it */
197:                    else if ( GOOD)
198:                    {
199:                            print_item ( item, sub,art_dir);
200:                            if(ncount == 0) printf("
                              not a artcle on
201:                            }
202:                    /* if just title, print all such titles */
203:                    else
204:                    {
205:                            for( i = 0; i < subjct_ct ; i++)
206:                            {
207:                                    if(subjcts[i]->excl == flag)
                                        continue;
208:                                    print_item ( item,
                                        subjcts[i]->name,art_dir);
209:                            }
210:                            if(ncount == 0) printf("There is no
                              artcle called
211:                    }
```

```
212:                    break;
213:
214:            default :
215:                    fprintf(stderr,"art:
                          option);
216:                    exit (1);
217:                    break;
218:            }
219: }
```

```
#include "art.h"
/*
*       main()
*       {
*       char fname[BUFSIZE], sub[BUFSIZE],
  f[BUFSIZE], comdir[BUFSIZE],tname[BUFSIZE];
*       int n;
*       strcpy(fname,
  "/usr/we/dock/ssubjects/test1/file1");
*       strcpy(sub,"test1");
*       strcpy(f,"file1");
*       strcpy(comdir,
  "/usr/we/dock/comments/test1/file1");
*       strcpy(tname,
  "/usr/we/dock/psubjects/test1/file1");
*       publish(fname,n,sub, f, comdir, tname);
*       }
*/
publish(fname,n,sub, f, comdir, tname, nm, newfname)
char *fname, *sub, *f, *comdir, *tname, *nm, *newfname;
int n;
{
int getpid();
char letter[BUFSIZE];
char mailinglist[BUFSIZE];
char message[BUFSIZE];
char sys[BUFSIZE];
char auth[10], *getlogin();
struct stat statinfo;
FILE *fopen(), *fp;

        sprintf(letter,"ltr%d",getpid());
        sprintf(message,"One file %s in SUBJECT %s was
          accepted        30:                    for publication", f, sub)

                /*      Place the information in file
                  letter    */
        fp = fopen(letter, "w");
        fputs(message,fp);
        fclose(fp);

                /*      Place info in mailing list      */
```

```
38:         sprintf(mailinglist,"mlg%d",getpid());
39:         if(stat(fname,&statinfo)==FALSE)
40:         {
41:                 printf(" %s is not a valid subject",fname)
                      ;
42:         }
43:         strcpy(auth,(getlogin(statinfo.st_uid)));
44:         fp = fopen(mailinglist, "w");
45:         fputs(auth,fp);
46:         fclose(fp);
47:
48:         /*  mail the info       */
49:         mail (letter,mailinglist);
50:         delete(letter);
51:         delete(mailinglist);
52:
53:
54:         /*      record for posterity    */
55:         sprintf(sys,"/usr/we/dock/record
              accepted for publication
56:         system(sys);
57:
58:         /*       move the file to new location  */
59:         sprintf(message,"mv %s %s", fname, tname);
60:         system(message);
61: }
62:
```

```
 1: /*
 2:  *   readdirs
 3:  *      reads all directories names and modification times
 4:  *      in the present working directory
 5:  *      allocates space as each entry is assigned to the
       structure
 6:  *      sorts entries in decreasing time order
 7:  *
 8:  *      arguments
 9:  *              file_ct         number of files found
10:  *              files           structure to fill
11:  *
12:  *      returns
13:  *              1   if cannot open "."
14:  *              0   otherwise
15:  */
16:
17: #
18: #include "art.h"
19:
20: char *ignored[] = {
21:         ".",
22:         "..",
23:         NULL
24: };
25:
26: readdirs (sub,file_ct,files)
27: char *sub;
28: int  *file_ct;
29: struct usersub **files;
30: {
31:         struct direct nf;
32:         struct stat sbuf;
33:         char fname[BUFSIZE];
34:         FILE *fd;
35:         int i, j;
36:         char *malloc(), *realloc();
37:
38:         if ((fd = fopen (sub,"r")) == NULL) {
39:                 return (1);
40:         }
41:
42:         /* Read the file names into files */
```

```
43:         *file_ct = 0;
44:         while (fread ((char *) &nf, sizeof nf, 1, fd) == 1)
              {
45:                 sprintf(fname,"%s/%s",sub,nf.d_name);
46:                 if (nf.d_ino != 0 && stat (fname, &sbuf) >=
                      0
47:                  && ((sbuf.st_mode & S_IFDIR) == S_IFDIR)) {
48:                         register char **p;
49:                         p = ignored;
50:                         while (*p && strncmp (*p, nf.d_name,
                               strlen(*p)))
51:                                 ++p;
52:                         if (!*p) {
53:                                 if ((*file_ct)++ > 0)
54:                                         *files = (struct
                                           usersub *)
55:                                                 realloc
                                                   ((char *) *files,
56:                                                 (unsigned)
57:                                                 (sizeof
                                                   (struct usersub)
58:                                                     * (*fil
                                                       e_ct)));
59:                                 else
60:                                         *files = (struct
                                           usersub *) malloc
61:                                                 ((unsigned)
62:                                                 (sizeof
                                                   (struct usersub) *
63:                                                 (*file_ct)))
                                                    ;
64:                                 if (*files == NULL) {
65:                                         fprintf (stderr,
                                             "No storage0);
66:                                         exit (1);
67:                                 }
68:                                 (*files)[(*file_ct)-1]
                                      .mtime = sbuf.st_mtime;
69:                                 strncpy ((*files)[(*file_ct)
                                      -1].name,
70:                                       .nf.d_name, DIRSIZ);
71:                         }
72:                 }
```

```
73:         }
74:
75:         /* Sort the elements of files in decreasing time
               order */
76:         for (i=1; i<(*file_ct); i++)
77:                 for (j=0; j<i; j++)
78:                         if ((*files)[j].mtime < (*files)[i]
                               .mtime) {
79:                                 struct usersub temp;
80:                                 temp = (*files)[i];
81:                                 (*files)[i] = (*files)[j];
82:                                 (*files)[j] = temp;
83:                         }
84:
85:         /* Clean up */
86:         fclose (fd);
87:         return (0);
88: }
```

```
 1: /*
 2:  *   readfiles
 3:  *       reads all regular file names and modification times
 4:  *       in the present working directory
 5:  *       allocates space as each entry is assigned to the
       structure
 6:  *       sorts entries in decreasing time order
 7:  *
 8:  *       arguments
 9:  *               file_ct         number of files found
10:  *               files           structure to fill
11:  *
12:  *       returns
13:  *               1   if cannot open "."
14:  *               0   otherwise
15:  */
16:
17: #
18: #include "art.h"
19:
20: char *ignore[] = {
21:         "core",
22:         NULL
23: };
24:
25: readfiles (sub,file_ct,files)
26: char *sub;
27: int  *file_ct;
28: struct usersub **files;
29: {
30:         struct direct nf;
31:         struct stat sbuf;
32:         char fname[BUFSIZE];
33:         FILE *fd;
34:         int i, j;
35:         char *malloc(), *realloc();
36:
37:         if ((fd = fopen (sub,"r")) == NULL) {
38:                 return (1);
39:         }
40:
41:         /* Read the file names into files */
42:         *file_ct = 0;
```

```
43:         while (fread ((char *) &nf, sizeof nf, 1, fd) == 1)
              {
44:                 sprintf(fname,"%s/%s",sub,nf.d_name);
45:                 if (nf.d_ino != 0 && stat (fname, &sbuf) >=
                      0
46:                  && (sbuf.st_mode & S_IFMT) == S_IFREG) {
47:                         register char **p;
48:                         p = ignore;
49:                         while (*p && strncmp (*p, nf.d_name,
                              DIRSIZ))
50:                                 ++p;
51:                         if (!*p) {
52:                                 if ((*file_ct)++ > 0)
53:                                         *files = (struct
                                              usersub *)
54:                                                 realloc
                                                      ((char *) *files,
55:                                                 (unsigned)
56:                                                 (sizeof
                                                      (struct usersub)
57:                                                       * (*fil
                                                            e_ct)));
58:                                 else
59:                                         *files = (struct
                                              usersub *) malloc
60:                                                 ((unsigned)
61:                                                 (sizeof
                                                      (struct usersub) *
62:                                                 (*file_ct)))
                                                      ;
63:                                 if (*files == NULL) {
64:                                         fprintf (stderr,
                                              "No storage0);
65:                                         exit (1);
66:                                 }
67:                                 (*files)[(*file_ct)-1]
                                      .mtime = sbuf.st_mtime;
68:                                 strncpy ((*files)[(*file_ct)
                                      -1].name,
69:                                         nf.d_name, DIRSIZ);
70:                         }
71:                 }
72:         }
```

```
 73:
 74:         /* Sort the elements of files in decreasing time
              order */
 75:         for (i=1; i<(*file_ct); i++)
 76:                 for (j=0; j<i; j++)
 77:                         if ((*files)[j].mtime < (*files)[i]
                               .mtime) {
 78:                                 struct usersub temp;
 79:                                 temp = (*files)[i];
 80:                                 (*files)[i] = (*files)[j];
 81:                                 (*files)[j] = temp;
 82:                         }
 83:
 84:         /* Clean up */
 85:         fclose (fd);
 86:         return (0);
 87: }
```

```
 1: #include "art.h"
 2:
 3: /*
 4:          *       main()
 5:          *       {
 6:          *       char fname[BUFSIZE], sub[BUFSIZE],
              f[BUFSIZE], comdir[BUFSIZE];
 7:          *       int n;
 8:          *       strcpy(fname,
              "/usr/we/dock/ssubjects/test1/file1");
 9:          *       strcpy(sub,"test1");
10:          *       strcpy(f,"file1");
11:          *       strcpy(comdir,
              "/usr/we/dock/comments/test1/file1");
12:          *
13:          *
14:          *       reject(fname,n,sub, f, comdir);
15:          *       }
16: */
17: reject(fname,n,sub, f, comdir, nm, newfname)
18: char *fname, *sub, *f, *comdir, *nm, *newfname;
19: int n;
20: {
21: int getpid();
22: char letter[BUFSIZE];
23: char mailinglist[BUFSIZE];
24: char message[BUFSIZE];
25: char dirnm[BUFSIZE];
26: char sys[BUFSIZE];
27: char date[BUFSIZE];
28: char auth[10], *getlogin();
29: struct stat statinfo;
30: FILE *fopen(), *fp;
31:
32:         sprintf(letter,"ltr%d",getpid());
33:         sprintf(message,"The file %s in SUBJECT %s was
              rejected0,
34:                 f, sub);
35:         /*      Place the information in file letter    */
36:         fp = fopen(letter, "w");
37:         fputs(message,fp);
38:         fclose(fp);
39:
```

```
40:
41:
42:         if(stat(fname,&statinfo)==FALSE)
43:         {
44:                 printf(" %s is not a valid subject",fname)
                      ;
45:         }
46:         sprintf(mailinglist,"ml%d",getpid());
47:         strcpy(auth,(getlogin(statinfo.st_uid)));
48:         fp = fopen(mailinglist, "a");
49:         fputs(auth,fp);
50:         fclose(fp);
51:         mail (letter,mailinglist);
52:
53:         sprintf(sys,"/usr/we/dock/record
              rejected article
54:         system(sys);
55:
56:
57:         delete(mailinglist);
58:         delete(letter);
59: }
```

```
 1:
 2:
 3: /*
 4:  *   report
 5:  *      prints artcle title. s
 6:  */
 7: report (s,sub,dummy,types)
 8:         char *s, *types,
 9:              sub[];
10: {
11:         static int first = 1;
12:         extern ncount;
13:
14:         if (s) {
15:                 if (first) {
16:                         first = 0;
17:                         switch (*types)
18:                         {
19:                         case 'r':printf("
                              revew:",sub);
20:                                 break;
21:                         case 's':printf("
                              submission:",sub);
22:                                 break;
23:                         default: printf ("
24:                         }
25:                         ncount = 5;
26:                 }
27:                 if(ncount%5 == 0) printf("      ");
28:                 printf ("%-15.14s", s);
29:                 ncount++;
30:         } else if (!first)
31:                 {
32:                         first = 1;
33:                         putchar ('0);
34:                 }
35: }
```

```
#include "art.h"
/*
*/
review(fname,n,sub, f, comdir, tname, nm, newfname)
char *fname, *sub, *f, *comdir, *tname, *nm, *newfname;
int n;
{
int getpid();
char letter[BUFSIZE];
char fcheck[BUFSIZE];
char mailinglist[BUFSIZE];
char message[BUFSIZE];
char dirnm[BUFSIZE];
char date[BUFSIZE];
char s[BUFSIZE];
char sys[BUFSIZE];
char auth[10], *getlogin();
struct stat statinfo;
FILE *fopen(), *fp;

        sprintf(letter,"ltr%d",getpid());
        sprintf(message,"Ohe file %s in SUBJECT %s was
          accepted        23:                    for review", f, sub);

        /*      Place the information in file letter    */
        fp = fopen(letter, "w");
        fputs(message,fp);
        fclose(fp);

        /*      Place info in mailing list      */
        sprintf(mailinglist,"mlg%d",getpid());
        if(stat(fname,&statinfo)==FALSE)
        {
                printf(" %s is not a valid subject",fname)
                  ;
        }
        strcpy(auth,(getlogin(statinfo.st_uid)));
        fp = fopen(mailinglist, "w");
        fputs(auth,fp);
        fclose(fp);

        /*  mail the info       */
```

```
42:         mail (letter,mailinglist);
43:         delete(letter);
44:         delete(mailinglist);
45:
46:
47:         /*      record for posterity    */
48:         sprintf(sys,"/usr/we/dock/artrecord
              accepted for review
49:         system(sys);
50:
51:         /*       move the file to new location  */
52:         sprintf(message,"mv %s %s", fname, tname);
53:         system(message);
54:
55:         /*      need a list of reviewers        */
56:         sprintf(s,"%scomments/%s/%s/reviewers",comdir,sub,f)
              ;
57:         sprintf(dirnm,"%scomments/%s",comdir,sub);
58:         sprintf(fcheck,"%s/members", dirnm);
59:         mklist(s,dirnm,fcheck);
60:
61:         /*      need to make directories for reviewers  */
62:         sprintf(message, "mkdirs %s/reviewers
              %scomments/%s/%s",newfname, comdir, sub, f);
63:         system(message);
64:
65:
66:         /*first is where to get list, second is dir to
              build them in    */
67:
68:         /*      Place the information in file letter    */
69:         sprintf(message,"Please add %s as a review subject",
              sub);
70:         fp = fopen(letter, "w");
71:         fputs(message,fp);
72:         fclose(fp);
73:         /*   mail directions to all reviewers   */
74:
75:
76:         /*  mail the info       */
77:         mail (letter,s);
78:
79: }
```

```
80:
```

```
 1:
 2: /*
 3:     submit.c
 4:
 5:         receives as arguments a filename (fname),
 6:         types, and an integer n which is the index
 7:         into the filename where one wishes to change
 8:         ?subject to comments.
 9:
10: */
11: #include "art.h"
12:
13: submit(fname,n,sub, f, comdir, nm, newfname)
14: char *fname, *sub, *f, *comdir, *nm, *newfname;
15: int n;
16: {
17:         char s[BUFSIZE];
18:         char tname[BUFSIZE];
19:         char NEWLINE;
20:         int answer;
21:
22:         printf( " Do you wish to accept for publication,")
              ;
23:         printf( " accept for review, or reject");
24:         for (answer=0; answer!=TRUE;)
25:         {
26:                 printf( " publish, review, or reject:");
27:                 s[0] = ' ';
28:                 if(gets(s) == NULL) exit(1);
29:                 switch (s[0])
30:                 {
31:                 case 'r':       if(strcmp(s,"rev") == 0 ||
32:                                 strcmp(s,"revi")   == 0 ||
33:                                 strcmp(s,"revie")  == 0 ||
34:                                 strcmp(s,"review") == 0)
35:                                 {
36:                                     strcpy(tname,fname);
37:                                     strncpy(&tname[n],"r",1)
                                          ;
38:                                     strcpy(s,"accepted for
                                          review");
```

```
39:                                     review(fname, n, sub, f,
                                           comdir,tname, nm, newfname);
40:                                     answer=TRUE;
41:                                     break;
42:                                     }
43:                                     if(strcmp(s,"rej") == 0 ||
44:                                     strcmp(s,"reje")   == 0 ||
45:                                     strcmp(s,"rejec")  == 0 ||
46:                                     strcmp(s,"reject") == 0)
47:                                     {
48:                                         reject(fname,n,sub, f,
                                              comdir, nm, newfname);
49:                                         answer=TRUE;
50:                                         break;
51:                                     }
52:                 case 'p':           if(strcmp(s,"p")   == 0 ||
53:                                     strcmp(s,"pu")     == 0 ||
54:
55:                                     strcmp(s,"pub")    == 0 ||
56:                                     strcmp(s,"publ")   == 0 ||
57:                                     strcmp(s,"publi")  == 0 ||
58:                                     strcmp(s,"publis") == 0 ||
59:                                     strcmp(s,"publish")== 0)
60:                                     {
61:                                         strcpy(tname,fname);
62:                                         strncpy(&tname[n],"p",1)
                                              :
63:                                         strcpy(s,"accepted for
                                              publishing");
64:                                         publish(fname, n, sub,
                                              f, comdir, tname, nm, newfname);
65:                                         answer=TRUE;
66:                                         break;
67:                                     }
68:
69:
70:                 case '?':           fprintf(stderr,"
                      %-10s%s0,"review","accepts the article for review by a list of reviewers
```

```
 71:                            fprintf(stderr,"
                                  %-10s%s0,"publish","accept for publication, can be vie
 72:                            fprintf(stderr,"
                                  %-10s%s0,"reject","reject the article, the author will
 73:                            break;
 74:
 75:
 76:                    default:        fprintf(stderr,"unknown
                          response, please try again.0);
 77:                            break;
 78:                    }
 79:            }
 80: }
 81:
 82:
 83:
```



```
  1:
  2: #include "art.h"
  3:
  4: typcheck(artdir,subject,title,types)
  5: char  *artdir, *subject, *title, *types;
```

```
 6: {
 7: char *getlogin(), nm[10], s[BUFSIZE],tmpnm[BUFSIZE],
      owner[10];
 8: int n, getpw();
 9: struct stat statinfo;
10:         strcpy(nm,getlogin());
11:         n=strlen(artdir) - 9;   /*size of string ssubjects*/
12:         strncpy(tmpnm,artdir,n);
13:         switch(*types)
14:         {
15:         case 's':       sprintf(s,"%srsubjects/%s",tmpnm,
              subject);
16:                         stat(s,&statinfo);
17:                         n=strlen(nm);
18:                         getpw(statinfo.st_uid,&owner[0]);
19:                         return(strncmp(owner,nm,n));
20:         case 'r':       sprintf(s,"%scomments/%s/%s/%s",
              tmpnm,subject,title,nm);
21:                         return(stat(s,&statinfo));
22:         default:        return(SUCCESS);
23:
24:         }
25: }
26:
27:
```

```
 1: /*
 2:  *   update
 3:  *      fwrites user_subs structure to file .user_subs
 4:  *      in user's HOME directory
 5:  *
 6:  *      arguments
 7:  *              sub             sub[s] to update
 8:  *              sub_ct  number of subs
 9:  *              user_subs       structure of user subs and
       times
10:  */
11:
12: #
13: #include "art.h"
14:
15: update (subjcts,subjct_ct)
16: struct subjcts *subjcts[MAXBDS];
17: int subjct_ct;
18: {
19:         struct usersub usersub;
20:         FILE *fopen(),
21:              *outfile;
22:         int  outcount;
23:         char *getenv(),
24:              *homeptr,
25:              home_subs[50];
26:         int  i;
27:
28:         if ( (homeptr = getenv("HOME")) == NULL )
29:         {
30:                 fprintf(stderr,"cannot find HOME
                    variable0);
31:                 exit(1);
32:         }
33:
34:
35:         strcpy (home_subs,homeptr);
36:         strcat (home_subs,"/");
37:         strcat (home_subs,".user_subs");
38:
39:         if((outfile = fopen (home_subs,"w")) == NULL)
40:         {
41:                 fprintf(stderr,"art: cannot open
                      .user_subs0);
```

```
42:                     exit(1);
43:             }
44:             for(i = 0; i < subjct_ct; i++)
45:             {
46:                     strcpy(usersub.name,subjcts[i]->name);
47:                     usersub.mtime = subjcts[i]->stbuf.st_mtime;
48:                     if( fwrite ( (char *) &usersub,
                          sizeof(struct usersub),1,outfile) != 1)
49:                     {
50:                             fprintf(stderr,"art: error writing
                                  .user_subs0);
51:                             exit (1);
52:                     }
53:             }
54:             fclose (outfile);
55: }
```

```
 1: /*
 2:  *   outartcls
 3:  *      processes artcl more current than cutoff
 4:  *
 5:  *      arguments
 6:  *              emit    artcle processing function
 7:  *                         typically print artcle or report
       title
 8:  *              sub     sub processed
 9:  *              cutoff  currency time, all artcl more
       current than
10:  *                         cutoff are "emitted"
11:  *
12:  *      assumes we are in artclsubs directory
13:  */
14:
15: #
16: #include "art.h"
17:
18: outartcls (emit,sub,art_dir,flag,types)
19: int  (*emit)();
20: struct subjcts *sub;
21: char *art_dir,*types;
22: int flag;
23: {
24:         struct usersub *artcls;
25:         int  artcl_ct,i;
26:         long cutoff;
27:         char fullname[BUFSIZE];
28:
29:         /* read artcle titles and modification times
                    */
30:         sprintf(fullname,"%s/%s",art_dir,sub->name);
31:         if( flag == EVERYONE)
32:         {
33:                 if (  readdirs (fullname, &artcl_ct,&artcls)
                     == 1 ) return ;
34:         }
35:         else
36:         {
37:                 if ( readfiles (fullname, &artcl_ct,&artcls)
                     == 1 ) return ;
38:         }
```

```
39:
40:         /* process current artcl
                        */
41:         for (i=0 ; i<artcl_ct; i++)
42:         {
43:                 if( (flag == ALL || flag == EVERYONE  ||
                      sub->new == 1 ||
44:                     artcls->mtime > sub->stbuf.st_mtime) &&
45:                     (typcheck(art_dir,sub->name,artcls->name,
                          types)==SUCCESS) )
46:                 {
47:                 (*emit) (artcls->name,sub->name,art_dir,
                      types);
48:                 }
49:                 artcls++;
50:         }
51:         (*emit) ((char *) NULL);
52:         fflush  (stdout);
53:
54: }
```

```
 1: #include <stdio.h>
 2: #include <sys/types.h>
 3: #include <sys/stat.h>
 4: #include <setjmp.h>
 5: #include <signal.h>
 6: #include <sys/dir.h>
 7: #include <pwd.h>
 8: #include <ctype.h>
 9:
10: #define BAD             (substat == subjct_ct)
11: #define NOTHERE         (substat == -1)
12: #define GOOD            (substat >= 0 && substat <
      subjct_ct)
13: #define EVERYONE        2
14: #define ALL             1
15: #define NEW             0
16: #define EXIST           0
17: #define SUCCESS         0
18: #define BOARD           *argv
19: #define A_BOARD         *(argv-1)
20: #define TITLE           *(argv-1)
21: #define OPTION          (*argv)[0] == '-' && (*argv)[2] ==
      NULL
22: /* The number of leading spaces on each line of output */
23: #define INDENT          3
24: #define BUFSIZE 256
25: #define MAXBDS 20
26: #define TRUE 1
27: #define FALSE -1
28:
29: /*
30:  *      The following items should not be printed.
31:  */
32: extern char *ignore[];
33:
34: struct usersub
35: {
36:         long mtime;
37:         char name[DIRSIZ];
38: };
39:
40: struct subjcts
41: {
```

```
42:         struct stat stbuf;
43:         int excl;
44:         int new;
45:         char name[DIRSIZ];
46: };
47:
48:
49: extern char stdbuf[BUFSIZ];
50:
51: extern jmp_buf  save_addr;
52:
53:
54: extern int ncount;
55: extern long time();
```

```
1:
2: if test ! -f $HOME/.artprofile
3: then
4:     cp /usrb/we/dock/src/artprofile.d $HOME/.artprofile
5:     cat /usrb/we/dock/src/profile.d >>$HOME/.profile
6: fi
7: /usrb/we/dock/bin/chgsubs
8: /usrb/we/dock/bin/chgtim
```

```
1:
2:
3: #artrecord "string to be printed" file */
4.
5:
6: DATE=`date | awk '{print $2,$3,$6}'`
7: echo "$DATE    $1" >>$2
```

```
Tue Jul 17 11:46 1984            artsubject.sh  33-1  /1

 1: #artsub.sh shell file
 2: #called artsub
 3:
 4: #DIR=`pwd`
 5: DEFDIR=/usr/we/dock
 6: cd $DEFDIR/comments
 7: SUBJECT=""
 8: while
 9:         test -z "$SUBJECT"
10: do
11:         echo "SUBJECT:        12:         read SUBJECT
13:
14:         if test -z "$SUBJECT"
15:         then
16:             echo "Enter the name of a subject."
17:             echo "Enter
18:             echo "of existing subjects."
19:             SUBJECT=""
20:             continue
21:         fi
22:
23:         if test  "$SUBJECT" = ?
24:         then
25:             echo "Enter the name of a subject."
26:             echo "The following are       27:              art +names +ever
28:             SUBJECT=""
29:             continue
30:         fi
31:         if test -d "$SUBJECT"
32:         then
33:             echo "$SUBJECT already exists as the name of a
                  subject."
34:             echo "Enter
35:             echo "of existing subjects."
36:             SUBJECT=""
37:             continue
38:         fi
39: done
40:
41: mkdir $SUBJECT        42: ../psubjects/$SUBJECT
```

```
../ssubjects/$SUBJECT
../rsubjects/$SUBJECT

#OWNER
OWNER=""
while
        test -z "$OWNER"
do
        echo "OWNER:
        read OWNER

        if test -z "$OWNER"
        then
            echo "Enter the login id of the owner."
            echo "Enter
            echo "of valid login ids."
            OWNER=""
            continue
        fi

        if test  "$OWNER" = ?
        then
            echo "Enter the login id of the owner."
            echo "The following are valid login ids
            OWNER       =""
            continue
        fi


        if  valid $OWNER
        then
            echo
        else
            echo "$OWNER is invalid name."
            echo "Enter
              idsO
            OWNER=""
            continue
        fi
done
chown $OWNER $DEFDIR/psubjects/$SUBJECT
```

```
 85: chown $OWNER $DEFDIR/rsubjects/$SUBJECT
 86:
 87: echo " A list of valid members of the subject must be
       created."
 88: echo " All valid reviewers of articles in this subject must
       be"
 89: echo " members of this group."
 90:
 91: /usr/we/dock/bin/mkmem
       $DEFDIR/comments/$SUBJECT/members  0
 92:
 93:
 94: ##cd DIR
```

```
 1: ARTDIR=/usr/we/dock
 2:
 3: # get user long name
 4: UNAME=`uname -n`
 5: MYNAME=`information - +me`
 6: if test -z "$MYNAME"
 7:     then MYNAME=${UNAME}!${LOGNAME}
 8: fi
 9:
10: # remember current directory
11: DIR=`pwd`
12:
13: cd $ARTDIR/comments
14:
15:
16: # get sub name and artcle title
17:
18: SUBJECT=$1
19: while
20:         test -z "$SUBJECT" -a -z "$ANSWER"
21: do
22:
23:         echo "SUBJECT:        24:         read SUBJECT
25:
26:         if test -z "$SUBJECT"
27:         then    echo "Enter the name of a sub."
28:                 echo "Enter
                      existing subs."
29:                 continue
30:         fi
31:
32:
33:         if test $SUBJECT = ?
34:         then    echo "Enter the name of a sub."
35:                 echo "The following are       36:                 $ARTDIR/bin/art +e
37:                 SUBJECT=""
38:                 continue
39:         fi
40:
41:
42:         if test ! -d $SUBJECT
```

```
43:         then    echo sub
44:                 echo "Enter
                      existing subs."
45:                 SUBJECT=""
46:         fi
47:
48: done
49:
50:
51: cd $SUBJECT
52: TITLE=$2
53:
54: while
55:
56:         test -z "$TITLE"
57:
58: do
59:         echo "TITLE:        60:         read TITLE
61:
62:         if test -z "$TITLE"
63:         then    echo "Enter the title for the artcle."
64:                 echo "Enter
                      artcles."
65:                 continue
66:         fi
67:
68:         # check for ? (help)
69:         if test "$TITLE" = ?
70:         then    echo "Enter a title for the artcle."
71:                 echo "The title may contain spaces. A
                      filename will be constructed from the title."
72:                 echo "The following titles (filenames)
                      already exist on       73:                  $ARTDIR/bin/art +o "$SUBJEC
74:                 TITLE=""
75:                 continue
76:         fi
77:
78:
79:         # create file name
80:         # check for existing name
81:
```



```
82:         FILE=`artfile $TITLE`
83:         OWNER=""
84:
85:         if test -z "$FILE"
86:         then    exit
87:         fi
88:
```

```
89:         if test ! "$TITLE" = "$FILE"
90:         then    echo Note:
              of artcle.
91:         fi
92:
93:         if test -f "$FILE"
94:         then    OWNER='ls -l $FILE | sed "s/  */:/g" | cut
              -f3 "-d:"'
95:                 if test ! -w $FILE -o ! "$OWNER" =
                      "$LOGNAME"
96:                 then    echo
                      you do not have permission to edit it.
97:                         TITLE=""
98:                 fi
99:         fi
100:
101:
102:
103: done
104:
105: trap "rm >/dev/null 2>/dev/null /tmp/artfile$$:exit" 1 2 15
106:
107:
108: cd $DIR
109:
110: mkdir $ARTDIR/comments/$SUBJECT/$FILE
111: /usr/we/dock/record "$MYNAME  submitted article for
       review" ""
112:
113: ORIGFILE=""
114: while
115:         test -z "$ORIGFILE"
116: do
117:
118:         echo "File to be copied:      119:          read ORIGFILE
```

```
120:
121:          if test -z "$ORIGFILE"
122:          then    echo "Enter the name of file to be copied."
123:                  echo "Enter
124:                  continue
125:          fi
126:
127:
128:          if test $ORIGFILE = ?
129:          then    echo "Enter the name of the file to be
                copied."
130:                  echo "it may be a full pathname or a
                        relative path name."
131:                  ORIGFILE=""
132:                  continue
133:          fi
134:
135:
136:          if test ! -f $ORIGFILE
137:          then    echo The file
138:                  echo "Enter
139:                  ORIGFILE=""
140:                  continue
141:          fi
142:
143: done
144:
145:
146: cp $ORIGFILE $ARTDIR/ssubjects/$SUBJECT/$FILE
```

```
 1:
 2: #arttrack   or
 3: #arttrack  SUBJECT   or
 4: #arttrack   SUBJECT ARETICLE
 5:
 6: cd /usrb/we/dock/comments
 7: MYNAME=`who am i| awk '{print $1}' `
 8: SUBJECT=$1
 9: ARTICLE=$2
10: ANS=""
11: VALID=""
12:
13: while
14:     test -z "$VALID"
15: do
16:     echo "Do  you  wish tc track articles for which you"
17:     echo "are the author or reviewer, or  a subject for"
18:     echo "which you are the owner ?"
19:     echo ; echo "type author, reviewer, or owner : "
20:
21:     read ANS
22:
23:     case "$ANS" in
24:         author| autho| auth| aut| au|a)
25:            VALID="OK"
26:             if test -z "$ARTICLE"
27:             then
28:                 if test -z "$SUBJECT"
29:                 then
30:                    for i in */*
31:                    do
32:                        if test -w $i -a -d $i
33:                        then
34:                            echo; echo "    SUBJECT/TITLE"
35:                            echo "    $i"; echo
36:                            cat $i/history
37:                            echo;echo
38:                        fi
39:                    done
40:                else
41:                    for i in $SUBJECT/*
42:                    do
43:                        if test -w $i -a -d $i
```

```
44:                         then
45:                             echo; echo "    SUBJECT/TITLE"
46:                             echo "    $i"; echo
47:                             cat $i/history
48:                             echo;echo
49:                         fi
50:                     done
51:                 fi
52:             else
53:              if test -w $SUBJECT/$ARTICLE
54:              then
55:                  echo; echo "    SUBJECT/TITLE"
56:                  echo "    $SUBJECT/$ARTICLE"; echo
57:                  cat $SUBJECT/$ARTICLE/history
58:                  echo;echo
59:              fi
60:         fi
61:
62:         ;;
63:         owner|owne|own|ow|o)
64:             VALID="OK"
65:             if test -z "$SUBJECT"
66:             then
67:                 for i in *
68:                 do
69:                     if test -w $i
70:                     then
71:                         cd $i
72:                         for j in *
73:                         do
74:                             if test -d $j
75:                             then
76:                                 echo; echo "
                                      SUBJECT/TITLE"
77:                                 echo "  $i/$j"; echo
78:                                 cat $j/history
79:                             fi
80:                         done
81:                         cd ..
82:                     fi
83:                 done
84:         else
85:             if test  -w $SUBJECT
```

```
 86:             then
 87:                  cd $SUBJECT
 88:                  for j in *
 89:                  do
 90:                       if test -d $j
 91:                       then
 92:                            echo; echo "    SUBJECT/TITLE"
 93:                            echo "  $SUBJECT/$j"; echo
 94:                            cat $j/history
 95:                       fi
 96:                  done
 97:             fi
 98:         fi
 99:         ;;
100:         reviewer|reviewe|review|revie|revi|rev|re|r)
101:             VALID="OK"
102:             if test -z      "$ARTICLE"
103:             then
104:                if test -z "$SUBJECT"
105:                then
106:                     for i in */*
107:                     do
108:                     cd  $i
109:                          for j in $MYNAME
110:                               do
111:                               echo; echo "    SUBJECT/TITLE"
112:                               echo "    $i":echo
113:                               egrep "submitted|accepted|$MYNA
                                    ME" history
114:                               echo ; echo
115:                               cd ../..
116:                          done
117:                     done
118:                else
119:                     for i in $SUBJECT/*
120:                     do
121:                     cd $i
122:                          for j in $MYNAME
123:                          do
124:                               echo; echo "    SUBJECT/TITLE"
125:                               echo "    $i":echo
126:                               egrep "submitted|accepted|$MYNA
                                    ME" history
```

```
127:                                echo ; echo
128:                                cd ../..
129:                            done
130:                        done
131:                    fi
132:                else
133:                    if  test !  -f $SUBJECT/$ARTICLE/$MYNAME
134:                    then
135:                        echo: echo "    SUBJECT/TITLE"
136:                        echo "    $SUBJECT/$ARTICLE":echo
137:                        cd $SUBJECT/$ARTICLE
138:                        egrep "submitted|accepted|$MYNAME"
                              history
139:                        echo ; echo
140:                        cd ../..
141:                    fi
142:                fi
143:            ::
144:        esac
145:
146:        if test -z "$VALID"
147:        then
148:            echo "$ANS is an invalid response"
149:        fi
150:    done
```

```
cp $HOME/.artprofile $HOME/.tmpartprofile
cd /usrb/we/dock/comments
for i in *
do
    if grep $i $HOME/.artprofile  >/dev/null
    then
        echo "$i is listed as a subject, do you wish to"
        echo "keep it in the list ? type yes or no. [yes]"
        read ans
        case "$ans" in
            no|n)
ed - $HOME/.tmpartprofile << !
g/ARTSUB/s/:$i//
w
!
            ;;
            yes|ye|y|*)
            ;;
        esac
    else
        echo "$i is a valid subject which is not"
        echo "listed in  your  subjects.  Do  you  wish"
        echo "to add the subject. Type yes or no. [yes]"
        read ans
        case "$ans" in
            yes|ye|y)
            sed -e "s/ARTSUBS=/&:$i/" $HOME/.tmpartprofile
              > $HOME/foop
            cp $HOME/foop $HOME/.tmpartprofile
            ;;
            no|n)
            ;;
            *)
            sed -e "s/ARTSUBS=/&:$i/" $HOME/.tmpartprofile
              > $HOME/foop
            cp $HOME/foop $HOME/.tmpartprofile
            ;;
        esac
    fi
done
cp $HOME/.tmpartprofile $HOME/.artprofile
```

```
 1: WHEN=`echo $ART_START`
 2: if  test -z "$WHEN"
 3: then
 4:         WHEN="1"
 5: fi
 6: echo "The start up time for ARTHER is set to execute  "
 7: case "$WHEN" in
 8: 1)
 9: echo "the first time you log on each day. Do you wish"
10: ;;
11: 2)
12: echo e="each time you log in.  Do you  wish"
13: ;;
14: 0)
15: echo "only on demand. Do you wish"
16: ;;
17: esac
18: echo " to change the time ? the choices are:"
19: echo "  (0)     only on demand"
20: echo "  (1)     first log on of the day"
21: echo "  (2)     each time you log on"
22: echo
23: echo "type 0, 1, 2, for the desired change"
24: echo "type a carriage return for no change."
25: read ans
26: if test ! -z "$ans"
27: then
28:    sed -e "s/ART_START=$WHEN/ART_START=$ans/"
         $HOME/.artprofile >$HOME/foop
29:    cp $HOME/foop $HOME/.artprofile
30: fi
```

```
 1: cp $HOME/.artprofile $HOME/.tmpartprofile
 2:     if grep $i $HOME/.artprofile  >/dev/null
 3:     then
 4:         echo "$i is listed as a subject. do you wish to"
 5:         echo "keep it in the list ? type yes or no. [yes]"
 6:         read ans
 7:         case "$ans" in
 8:             no|n)
 9: ed - $HOME/.tmpartprofile << !
10: g/ARTSUB/s/:$i//
11: w
12: !
13:             ;;
14:             yes|ye|y|*)
15:             ;;
16:         esac
17:     else
18:         echo "$i is a valid subject which is not"
19:         echo "listed in  your  subjects.  Do  you  wish"
20:         echo "to add the subject. Type yes or no. [yes]"
21:         read ans
22:         case "$ans" in
23:             yes|ye|y)
24:             sed -e "s/ARTSUBS=/&:$i/" $HOME/.tmpartprofile
                  > $HOME/foop
25:             cp $HOME/foop $HOME/.tmpartprofile
26:             ;;
27:             no|n)
28:             ;;
29:         esac
30:     fi
31: done
32: cp $HOME/.tmpartprofile $HOME/.artprofile
```

```
 1:
 2: ANS=""
 3:
 4: echo here
 5: sed  -n  '1p' try
 6: while
 7:          test -z "$ANS"
 8: do
 9:          echo "  keep or remove, keep? "
10:          read ANS
11:          if test "$ANS" = keep
12:
13:          then
14:                    sed -n d
15:          else
16:                    sed -n  'p
17:          fi
18: done
19:
```

```
1:
2: for i in `cat $1`
3: do
4: echo $2/$i
5: mkdir $2/$i
6: done
```

```
1:
2: /bin/mail `cat $1` < $2
3:
```

```
 1: SRCDIR=`pwd`
 2: cd ..
 3: NEWARTDIR=`pwd`
 4: cd $SRCDIR
 5: OLDARTDIR="/usrb/we/dock"
 6: replace $OLDARTDIR $NEWARTDIR *
 7: cd ..
 8: mkdir comments
 9: mkdir ssubjects
10: mkdir psubjects
11: mkdir rsubjects
12: mkdir bin
13:
```

```
 1:
 2: if test -d  /usr/we/dock/comments/$1
 3: then
 4:         rm    ?subjects/$1/* >/dev/null 2>&1
 5:         rmdir ?subjects/$1 >/dev/null 2>&1
 6:         rm    comments/$1/*/*/* >/dev/null 2>&1
 7:         rm    comments/$1/*/* >/dev/null 2>&1
 8:         rm    comments/$1/* >/dev/null 2>&1
 9:         rmdir comments/$1/*/* >/dev/null 2>&1
10:         rmdir comments/$1/* >/dev/null 2>&1
11:         rmdir comments/$1 >/dev/null 2>&1
12: else
13:         echo "  $1 is an invalid subject."
14:         art +g
15:         exit
16: fi
```

```
1:
2: grep "^$1" $2 | exit 0
3: exit 1
```

```
ARTBIN =  /usrb/we/dock
CC = cc



ARTOBJS =       art.o process.o given_sub.o get_subs.o outart
                onintr.o output.o initialize.o delete.o edi
                mklist.o publish.o reject.o review.o typchec
                listcreate.o readfiles.o

install: art  artfile mklist mkmem comment
        sh -x xdock

art : $(ARTOBJS)
        $(CC) -o art $(ARTOBJS)
        chmod 0755 art
        cp art ../bin/art

artsubject :    artfile.o
        cp artsubject.sh artsubject
        chmod 0755 artsubject


artfile.o :
        ${CC} -O artfile.c -o  artfile
        chmod 0755 artfile
        cp artfile ../bin/artfile

mklist :  mklist.o mklistcom.o edit.o listcreate.o
        ${CC} -O mklist.o mklistcom.o listcreate.o edit.o   -o mklist
        chmod 0755 mklist
        cp mklist ../bin/mklist

mkmem : art.h mkmem.o listcreate.o edit.o
        ${CC} mkmem.o listcreate.o edit.o -o mkmem
        chmod 0755 mkmem
```

```
41:         cp mkmem ../bin/mkmem
42:
43:
44: comment : comment.o
45:         ${CC} -O comment.o   -o artcomment
46:         chmod 0755 artcomment
47:         cp artcomment ../bin/artcomment
48:
49:
50: clean :
51:         -rm -f *.o
52:         -rm -f artsubject
53:
54: lpr :
55:         -xpr -!  Makefile art.h *.c *.sh | lpr
```

A p p e n d i x 4

Application Code Cross References

```
Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS      1

          cominfo     3-   4 cominfo.c
          delete      5-  10 delete.c
          edit        6-   9 edit.c
          get_subs    7-  17 get_subs.c
          given_sub   8-  14 given_sub.c
          initializ   9-   7 initialize.c
          listcreat  10-   9 listcreate.c
          mail       11-   2 mail.c
          main        1-  31 art.c
          main        2-   9 artfile.c
          main        4-  16 comment.c
          main       13-   5 mklistcom.c
          main       14-   5 mkmem.c
          mklist     12-   3 mklist.c
          newcount   15-   5 newcount.c
          onintr     16-   5 onintr.c
          outartcls  29-  18 uutsubs.c
          output     17-   1 output.c
          print_ite  18-   6 print_item.c
          process    19-   4 process.c
          publish    20-  15 publish.c
          readdirs   21-  26 readdirs.c
          readfiles  22-  25 readfiles.c
          reject     23-  17 reject.c
          report     24-   7 report.c
          review     25-   4 review.c
          submit     26-  13 submit.c
          typcheck   27-   4 typcheck.c
          update     28-  15 update.c

_exit     initializ   9-  10 extern _exit();
                      9-  13 signal (SIGQUIT, _exit);

chdir     cominfo     3-  12 chdir(s);

cominfo   #DEFN       3-   4 <<DEFN>> cominfo(dirnm,nm)
          print_ite  18-  48 cominfo(newfname,nm);

count     process    19-  11 int  print_item(), report(), count()
                                , newcount();

ctime     print_ite  18-  32 char *ctime();



Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS      2  /delete

delete    #DEFN        5-  10 <<DEFN>> delete(fname)
          publish     20-  50 delete(letter);
                      20-  51 delete(mailinglist);
          reject      23-  57 delete(mailinglist);
                      23-  58 delete(letter);
```

```
          review     25-  43 delete(letter);
                     25-  44 delete(mailinglist);

edit      #DEFN       6-   9 <<DEFN>> edit(fname)
          main       14-  19 edit(argv[1]);
                     14-  25 edit(argv[1]);
          mklist     12-  58 edit(fname);

exit      get_subs    7-  37 exit(1);
                      7-  43 exit(1);
          main        1- 103 if (runcmd == NO) exit(1);
                      1- 119 exit();
                      2-  32 if(++k == 14) exit(0);
                     13-  25 if (gets(s) == NULL) exit(1);
                     13-  62 exit(0);
                     14-  22 case 'q':   exit(1);
          print_ite  18-  67 exit(1);
          process    19- 216 exit (1);
          readdirs   21-  66 exit (1);
          readfiles  22-  65 exit (1);
          submit     26-  28 if(gets(s) == NULL) exit(1);
          update     28-  31 exit(1);
                     28-  42 exit(1);
                     28-  51 exit (1);

fclose    edit        6-  41 fclose(fp);
                      6-  42 fclose(tmp);
          listcreat  10-  54 fclose(fp);
          main        4-  35 fclose(fp);
          print_ite  18-  52 fclose (fd);
          publish    20-  35 fclose(fp);
                     20-  46 fclose(fp);
          readdirs   21-  86 fclose (fd);
          readfiles  22-  85 fclose (fd);
          reject     23-  38 fclose(fp);
                     23-  50 fclose(fp);
          review     25-  28 fclose(fp);
                     25-  39 fclose(fp);
```

```
                    25-  72 fclose(fp);
          update    28-  54 fclose (outfile);

fflush    outartcls 29-  52 fflush  (stdout);

fgets     edit       6-  26 while(fgets(s, sizeof s,fp)!=NULL)

fileno    print_ite 18-  37 fstat (fileno (fd), &sbuf);

fopen     edit       6-  15 FILE  *fopen(), *tmp, *fp;
                      6-  20 if((tmp=fopen(tmpfile,"w"))== NULL)

                      6-  22 if((fp=fopen(fname,"r"))== NULL)
          get_subs    7-  27 FILE *fopen(), *infile;
          histcreat  10-  12 FILE *fopen(), *fp;
                     10-  17 if ((fp=fopen(fname,"a"))== NULL)
          main        4-  20 FILE *fopen(), *fp;
                      4-  22 fp=fopen("comment","a");
          print_ite  18-  28 if ((fd = fopen (fname, "r")) !=
                                 NULL)
          publish    20-  26 FILE *fopen(), *fp;
                     20-  33 fp = fopen(letter, "w");
                     20-  44 fp = fopen(mailinglist, "w");
          readdirs   21-  38 if ((fd = fopen (sub,"r")) == NULL)
                                 {
          readfiles  22-  37 if ((fd = fopen (sub,"r")) == NULL)
                                 {
          reject     23-  30 FILE *fopen(), *fp;
                     23-  36 fp = fopen(letter, "w");
                     23-  48 fp = fopen(mailinglist, "a");
          review     25-  19 FILE *fopen(), *fp;
                     25-  26 fp = fopen(letter, "w");
                     25-  37 fp = fopen(mailinglist, "w");
                     25-  70 fp = fopen(letter, "w");
          update     28-  20 FILE *fopen(),
                     28-  39 if((outfile = fopen (home_subs,"w"))
                                 == NULL)

fprintf   get_subs    7-  36 fprintf(stderr,"art:
                                 name for ARTDIRO,art_dir);
                      7-  42 fprintf(stderr,"art: cannot read

          onintr     16-   7 fprintf(stderr,"Ointerrupt0);
```



```
          print_ite  18-  57 fprintf(stderr,"review, next, quit
                                 ? [next] ");
                     18-  70 case '?':       fprintf(stderr,"
                                 %-10s%s0,"review","print artcle
                     18-  71 fprintf(stderr,"   %-10s%s0,
```

```
                    "next","go to next article");
              18-  72 fprintf(stderr,"   %-10s%s0,
                    "quit","terminate program");
              18-  78 default:        fprintf(stderr,
                    "unknown command, please try
process       19-  31 if(ncount == 0) fprintf(stdout,
                    "There are no new artcl on
              19-  43 fprintf(stderr,"
                    existing sub.0,sub);
              19-  56 if(ncount == 0) fprintf(stdout,
                    "There are no artcl on
              19-  68 fprintf(stderr,"
                    existing sub.0,sub);
              19-  81 if(ncount == 0) fprintf(stdout,
                    "There are no artcl on
              19-  92 fprintf(stderr,"
                    existing sub.0,sub);
              19- 116 fprintf(stderr,"
                    existing sub.0,sub);
              19- 119 if(ncount > 0) fprintf(stdout,
                    "%d0,ncount);
              19- 132 if(ncount == 0) fprintf(stdout,
                    "There are no artcl on
              19- 146 fprintf(stderr,"
                    existing sub.0,sub);
              19- 155 fprintf(stdout,"
                    sub);
              19- 158 fprintf(stdout,"existing subjects
                    in your environment:");
              19- 164 fprintf(stdout,"%-15.14s",subjcts[i]
                    ->name);
              19- 170 fprintf(stdout,"existing subjects
                    :");
              19- 175 fprintf(stdout,"%-15.14s",subjcts[i]
                    ->name);
              19- 180 fprintf(stdout,"
                    0,sub);
              19- 215 fprintf(stderr,"art:
                    option0,option);
```

```
Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS     5  /fprintf

          readdirs   21-  65 fprintf (stderr, "No storage0);
          readfiles  22-  64 fprintf (stderr, "No storage0);
          submit     26-  70 case '?':        fprintf(stderr,"
                              %-10s%s0,"review","accepts the
                     26-  71 fprintf(stderr,"   %-10s%s0,
                              "publish","accept for publication,
                     26-  72 fprintf(stderr,"   %-10s%s0,
                              "reject","reject the article, the author
                     26-  76 default:         fprintf(stderr,
                              "unknown response, please try again.0);
          update     28-  30 fprintf(stderr,"cannot find HOME
                              variable0);
                     28-  41 fprintf(stderr,"art: cannot open
                              .user_subs0);
                     28-  50 fprintf(stderr,"art: error writing
                              .user_subs0);

fputs     edit        6-  37 case 'k':         fputs(s,tmp);
                      6-  38 fputs("0,tmp);
          listcreat  10-  46 fputs(s,fp);
                     10-  47 fputs("0,fp);
          main        4-  31 fputs(s,fp);
                      4-  32 fputs('0,fp);
          publish    20-  34 fputs(message,fp);
                     20-  45 fputs(auth,fp);
          reject     23-  37 fputs(message,fp);
                     23-  49 fputs(auth,fp);
          review     25-  27 fputs(message,fp);
                     25-  38 fputs(auth,fp);
                     25-  71 fputs(message,fp);

fread     readdirs   21-  44 while (fread ((char *) &nf, sizeof
                              nf, 1, fd) == 1) {
          readfiles  22-  43 while (fread ((char *) &nf, sizeof
                              nf, 1, fd) == 1) {

fstat     print_ite  18-  37 fstat (fileno (fd), &sbuf);

fwrite    update     28-  48 if( fwrite ( (char *) &usersub,
                              sizeof(struct usersub),1,outfile) != 1)

get_subs  #DEFN       7-  17 <<DEFN>> get_subs (art_dir,art_subs,
                              art_excl,subjcts,subjct_ct)



Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS     6  /get_subs

          main        1- 109 get_subs (art_dir,art_subs,art_excl,
                              subjcts,&subjct_ct);

getenv    initializ   9-  11 char *env,*getenv();
                      9-  26 env = getenv("ARTSUBS");
```

```
          update      28-  23 char *getenv(),
                      28-  28 if ( (homeptr = getenv("HOME")) ==
                                  NULL )

geteuid   main        13-  16 whoami=geteuid();

getlogin  cominfo      3-   7 char *getlogin(),  s[BUFSIZE],
                                  sys[BUFSIZE];
                       3-  10 strcpy(nm,getlogin());
          publish     20-  24 char auth[10], *getlogin();
                      20-  43 strcpy(auth,(getlogin(statinfo.st_u
                                  id)));
          reject      23-  28 char auth[10], *getlogin();
                      23-  47 strcpy(auth,(getlogin(statinfo.st_u
                                  id)));
          review      25-  17 char auth[10], *getlogin();
                      25-  36 strcpy(auth,(getlogin(statinfo.st_u
                                  id)));
          typcheck    27-   7 char *getlogin(), nm[10], s[BUFSIZE]
                                  ,tmpnm[BUFSIZE], owner[10];
                      27-  10 strcpy(nm,getlogin());

getpid    edit         6-  16 int getpid();
                       6-  19 sprintf(tmpfile, "tmp%d", getpid());

          publish     20-  19 int getpid();
                      20-  28 sprintf(letter,"ltr%d",getpid());
                      20-  38 sprintf(mailinglist,"mlg%d",getpid()
                                  );
          reject      23-  21 int getpid();
                      23-  32 sprintf(letter,"ltr%d",getpid());
                      23-  46 sprintf(mailinglist,"ml%d",getpid())
                                  ;
          review      25-   8 int getpid();
                      25-  21 sprintf(letter,"ltr%d",getpid());
                      25-  31 sprintf(mailinglist,"mlg%d",getpid()
                                  );
```

```
Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     7  /getpw

getpw       typcheck    27-   8 int n, getpw();
                        27-  18 getpw(statinfo.st_uid,&owner[0]);

gets        cominfo      3-  24 gets(s);
            edit         6-  29 gets(ans);
                         6-  35 gets(ans);
            listcreat   10-  26 gets(s);
                        10-  36 gets(s);
                        10-  52 gets(s);
            main         4-  29 for (gets(s); s[0] != '.' && s[1]
                                    != ' ';)
                         4-  33 gets(s);
                        13-  25 if (gets(s) == NULL) exit(1);
                        13-  53 gets(s);
                        13-  58 gets(s);
                        13-  65 gets(s);
                        14-  15 gets(s);
            mklist      12-  24 gets(s);
                        12-  28 gets(s);
                        12-  56 gets(s);
            print_ite   18-  58 gets(s);
            submit      26-  28 if(gets(s) == NULL) exit(1);

given_sub   #DEFN        8-  14 <<DEFN>> given_sub (sub,subjcts,
                                    subjct_ct)
            main         1- 146 substat = given_sub(sdopt,subjcts,
                                    subjct_ct);
            process     19-  27 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19-  52 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19-  77 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19- 101 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19- 126 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19- 153 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19- 184 substat = given_sub(sub,subjcts,
                                    subjct_ct);
                        19- 185 newsstat = given_sub(item,subjcts,
                                    subjct_ct);


Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     8  /initializ

initializ   #DEFN        9-   7 <<DEFN>> initialize(art_dir,
                                    art_subs,art_opts,art_excl,
            main         1- 108 initialize (art_dir,art_subs,
                                    art_opts,art_excl,ptypes,option);
```

```
isalnum   main         2-  24 if(isalnum(argv[i][j]) ||

isupper   process     19-  16 if( isupper(option) )

listcreat #DEFN       10-   9 <<DEFN>> listcreate(fname,fcheck)
          main        14-  12 listcreate(argv[1], "/etc/passwd");

          mklist      12-  48 listcreate(fname,fcheck);

longjmp   onintr      16-   8 longjmp(save_addr, 1);

mail      #DEFN       11-   2 <<DEFN>> mail(letter,mailinglist)
          publish     20-  49 mail (letter,mailinglist);
          reject      23-  51 mail (letter,mailinglist);
          review      25-  42 mail (letter,mailinglist);
                      25-  77 mail (letter,s);

main      #DEFN        1-  31 <<DEFN>> main (argc,argv)
                       2-   9 <<DEFN>> main(argc,argv)
                       4-  16 <<DEFN>> main()
                      13-   5 <<DEFN>> main()
                      14-   5 <<DEFN>> main(argc,argv)

malloc    get_subs     7-  26 char *malloc();
                       7-  58 subjcts[*subjct_ct] = malloc(
                                 sizeof (struct subjcts) );
          readdirs    21-  36 char *malloc(), *realloc();
          readfiles   22-  35 char *malloc(), *realloc();

mklist    #DEFN       12-   3 <<DEFN>> mklist(fname,dirname,
                                 fcheck)
          main        13-  67 mklist(fname,dirname, "/etc/passwd")
                                 ;
          review      25-  59 mklist(s,dirnm,fcheck);

newcount  #DEFN       15-   5 <<DEFN>> newcount (s,sub)
          process     19-  11 int  print_item(), report(), count()
                                 , newcount();
```

```
Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     9  /newcount

                    19- 104 outartcls(newcount.subjcts[substat],
                                art_dir,NEW,types);
                    19- 111 outartcls(newcount.subjcts[i],
                                art_dir,NEW,types);

onintr      #DEFN       16-   5 <<DEFN>> onintr()
            print_ite   18-  18 int onintr();
                        18-  45 signal(SIGINT, onintr);

open        get_subs     7-  40 if((fd = open(art_dir,0)) == -1)

outartcls   #DEFN       29-  18 <<DEFN>> outartcls (emit,sub,
                                art_dir,flag,types)
            main         1- 115 outartcls(print_item,subjcts[i],
                                art_dir,NEW,ptypes);
            process     19-  30 outartcls(report,subjcts[substat],
                                art_dir,NEW,types);
                        19-  38 outartcls(report,subjcts[i],art_dir,
                                NEW,types);
                        19-  55 outartcls(report,subjcts[substat],
                                art_dir,ALL,types);
                        19-  63 outartcls(report,subjcts[i],art_dir,
                                ALL,types);
                        19-  80 outartcls(report,subjcts[substat],
                                art_dir,EVERYONE,types);
                        19-  87 outartcls(report,subjcts[i],art_dir,
                                EVERYONE,types);
                        19- 104 outartcls(newcount,subjcts[substat],
                                art_dir,NEW,types);
                        19- 111 outartcls(newcount,subjcts[i],
                                art_dir,NEW,types);
                        19- 129 outartcls(print_item,subjcts[substa
                                t],art_dir,ALL,types);
                        19- 139 outartcls(print_item,subjcts[i],
                                art_dir,ALL,types);
                        19- 191 outartcls(print_item,subjcts[substa
                                t],art_dir,NEW,types);

output      #DEFN       17-   1 <<DEFN>> output(filename,outs)
            print_ite   18-  49 output(fname,&s[0]);

print_ite   #DEFN       18-   6 <<DEFN>> print_item (f,sub,art_dir,
                                ptypes)



Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     10 /print_ite

            main         1-  38 int print_item();
                         1- 115 outartcls(print_item,subjcts[i],
                                art_dir,NEW,ptypes);
            process     19-  11 int  print_item(), report(), count()
                                , newcount();
```

```
                    19- 129 outartcls(print_item,subjcts[substa
                              t],art_dir,ALL,types);
                    19- 139 outartcls(print_item,subjcts[i],
                              art_dir,ALL,types);
                    19- 191 outartcls(print_item,subjcts[substa
                              t],art_dir,NEW,types);
                    19- 199 print_item ( item, sub.art_dir);
                    19- 208 print_item ( item, subjcts[i]->name,
                              art_dir);

printf    cominfo    3-  15 printf("
                              to  review  the article.");
                     3-  16 printf("   You  will  be  unable
                              to  write  the   file.");
                     3-  17 printf("   To  enter  comments
                              for the  author  to read.");
                     3-  18 printf("   simply type

                     3-  19 printf("   comments.  REMEMBER:
                              the author will know what ");
                     3-  20 printf("   article you are
                              commenting upon, but not where");
                     3-  21 printf("    you are in the
                              program.   To exit the article");
                     3-  22 printf("   simply type
                     3-  23 printf("   Type a carriage
                              return when ready for the file.");
          delete     5-  14 printf(" unable to remove list.");

          edit       6-  21 printf("Onable to open %s",
                              tmpfile);
                     6-  23 printf("Onable to open %s",fname);

                     6-  28 printf("95s: keep or delete
                              [keep]?", s);
                     6-  32 case '?':       printf (" keep:.
                              entry  remains in  list");
                     6-  33 printf (" delete: removes entry
                              from list");
```

```
                    6-  34 printf("%6s: keep or delete
                           [keep]?", s);
          listcreat 10-  18 printf(" can't open fname");
                    10-  20 printf(" Enter  names  one  at  a
                           time");
                    10-  21 printf(" when finished, type  a
                           period");
                    10-  22 printf(" followed by a carriage
                           return.");
                    10-  23 printf(" For help type a question
                           mark.");
                    10-  25 printf(" name: ");
                    10-  32 printf(" Enter  names  one  at  a
                           time");
                    10-  33 printf(" when finished, type  a
                           period");
                    10-  34 printf(" followed by a carriage
                           return.");
                    10-  35 printf(" name: ");
                    10-  50 printf(" invalid entry");
                    10-  51 printf(" name: ");
          main       1-  79 printf(ILL,argv[i] , USE);
                     1-  87 printf(ILLTOG, ptypes, tempt, USE);

                     1-  95 printf(ILLTOG, ptypes, tempt, USE);

                     4-  24 printf( " Enter your comments a
                           line at a time.");
                     4-  25 printf( " When you have completed
                           your comment,");
                     4-  26 printf( " begin  a  line  with  a
                           period   (.)");
                     4-  27 printf( " followed by cariage
                           return.");
                     4-  28 printf( " ");
                    13-  19 printf(" Lists are created on a
                           subect basis by the owner of the group.");
                    13-  22 printf(" subject: ");
                    13-  29 printf("\nter the name of the
                           subject for which you wish to");
                    13-  30 printf(" create the list. You
                           must be the owner of the group");
                    13-  31 printf(" Type quit to exit.");
```



```
                    13-  38 printf(" %s is not a valid
                           subject",s);
                    13-  46 printf(" You are not the owner of
                           the group %s",s);
                    13-  50 printf(" Do you wish to create a
```

```
                    new list. delete an existing list.");
          13-  51 printf(" or edit an existing
                    list?");
          13-  52 printf(" create. delete. or edit?
                    ");
          13-  57 printf(" name of existing list: ")
                    ;
          13-  64 printf(" Name of the list you
                    wish to create or edit? ");
          14-  14 {       printf("Do you wish to
                    edit the file or quit? ");
          14-  23 case '?':   printf("Odit: examine
                    the entries one at a time");
          14-  24 printf("Quit: leave the list as
                    it is.");
mklist    12-  21 printf(" The list you have
                    indicated does not");
          12-  22 printf("exist. Do you wish to
                    duplicate an existing");
          12-  23 printf("list for this suartject ? ")
                    ;
          12-  27 case 'y':       printf(" list: ");

          12-  55 printf(" Do you wish to edit the
                    file or quit?");
print_ite 18-  47 printf("   subject: %s
                    article: %s0.sub,f);
process   19- 163 if((j++)%5 == 0) printf("      ");

          19- 174 if((j++)%5 == 0) printf("      ");

          19- 194 if(ncount == 0) printf("There are
                    no new artcl on
          19- 200 if(ncount == 0) printf("
                    not a artcle on
          19- 210 if(ncount == 0) printf("There is no
                    artcle called
publish   20-  41 printf(" %s is not a valid
                    subject",fname);
```

```
Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS      13 /printf

          reject      23-  44 printf(" %s is not a valid
                             subject",fname);
          report      24-  19 case 'r':printf("
                             revew:",sub);
                      24-  21 case 's':printf("
                             submission:",sub);
                      24-  23 default: printf ("

                      24-  27 if(ncount%5 == 0) printf("      ");

                      24-  28 printf ("%-15.14s", s);
          review      25-  34 printf(" %s is not a valid
                             subject",fname);
          submit      26-  22 printf( " Do you wish to accept
                             for publication.");
                      26-  23 printf( " accept for review, or
                             reject");
                      26-  26 printf( " publish, review, or
                             reject:");

process   #DEFN       19-   4 <<DEFN>> process(subjcts,subjct_ct,
                             art_dir,sub,item,option,types)
          main         1- 130 process(subjcts,subjct_ct,art_dir,
                             "","",option,ptypes);
                       1- 136 process(subjcts,subjct_ct,art_dir,
                             sdopt,"",option,ptypes);
                       1- 151 process(subjcts,subjct_ct,art_dir,
                             sub,"",' ',ptypes);
                       1- 154 process(subjcts,subjct_ct,art_dir,
                             sub,sdopt,' ',ptypes);

publish   #DEFN       20-  15 <<DEFN>> publish(fname,n,sub, f,
                             comdir, tname, nm, newfname)
          submit      26-  64 publish(fname, n, sub, f, comdir,
                             tname, nm, newfname);

putc      main         2-  31 putc(argv[i][j],stdout);
                       2-  38 putc('_',stdout);
                       2-  41 putc('0,stdout);
          print_ite   18-  62 putc('0,stdout);
                      18-  82 putc('0,stdout);
          process     19- 166 putc('0,stdout);
                      19- 177 putc('0,stdout);



Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS      14 /putchar

putchar   print_ite   18-  40 putchar ('0);
                      18-  51 putchar ('0);
          report      24-  33 putchar ('0);

read      get_subs     7-  48 while(read(fd,(char *)&dirbuf,
```

```
                        sizeof(dirbuf))>0)

readdirs  #DEFN      21-  26 <<DEFN>> readdirs (sub,file_ct,
                            files)
          outartcls  29-  33 if (  readdirs (fullname, &artcl_ct,
                            &artcls) == 1 ) return ;

readfiles #DEFN      22-  25 <<DEFN>> readfiles (sub,file_ct,
                            files)
          outartcls  29-  37 if ( readfiles (fullname, &artcl_ct,
                            &artcls) == 1 ) return ;

realloc   readdirs   21-  36 char *malloc(), *realloc();
                     21-  55 realloc ((char *) *files,
          readfiles  22-  35 char *malloc(), *realloc();
                     22-  54 realloc ((char *) *files,

reject    #DEFN      23-  17 <<DEFN>> reject(fname,n,sub, f,
                            comdir, nm, newfname)
          submit     26-  48 reject(fname,n,sub, f, comdir, nm,
                            newfname);

report    #DEFN      24-   7 <<DEFN>> report (s,sub,dummy,types)

          process    19-  11 int  print_item(), report(), count()
                            , newcount();
                     19-  30 outartcls(report,subjcts[substat],
                            art_dir,NEW,types);
                     19-  38 outartcls(report,subjcts[i],art_dir,
                            NEW,types);
                     19-  55 outartcls(report,subjcts[substat],
                            art_dir,ALL,types);
                     19-  63 outartcls(report,subjcts[i],art_dir,
                            ALL,types);
                     19-  80 outartcls(report,subjcts[substat],
                            art_dir,EVERYONE,types);
                     19-  87 outartcls(report,subjcts[i],art_dir,
                            EVERYONE,types);
```

```
Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     15 /review

review     #DEFN       25-   4 <<DEFN>> review(fname,n,sub, f,
                                 comdir, tname, nm, newfname)
           submit      26-  39 review(fname, n, sub, f, comdir,
                                 tname, nm, newfname);

setjmp     print_ite   18-  42 if (setjmp(save_addr))

signal     initializ    9-  12 if (signal (SIGQUIT, SIG_IGN) !=
                                 SIG_IGN)
                        9-  13 signal (SIGQUIT, _exit);
           print_ite   18-  44 if (signal(SIGINT, SIG_IGN) !=
                                 SIG_IGN)
                       18-  45 signal(SIGINT, onintr);
                       18-  53 if (signal(SIGINT, SIG_IGN) !=
                                 SIG_IGN)
                       18-  54 signal(SIGINT, SIG_DFL);

sprintf    cominfo      3-  11 sprintf(s,"%s/%s",dirnm,nm);
                        3-  13 sprintf(sys,"/usr/we/dock/ar
                                 trecord
           edit         6-  19 sprintf(tmpfile, "tmp%d", getpid());

                        6-  43 sprintf(s,"mv %s %s",tmpfile, fname)
                                 ;
           get_subs     7-  54 sprintf(fullname,"%s/%s",art_dir,
                                 dirbuf.d_name);
           listcreat   10-  43 sprintf(sys, "/usr/we/dock/v
                                 alid %s %s",s,fcheck);
           mail        11-   6 sprintf(com,"/usr/we/dock/ml
                                 %s %s",mailinglist, letter);
           main         1-  71 sprintf(subnames + strlen(subnames),
                                 " %s",argv[i]);
                       13-  35 sprintf(dirname,"%s/comments/%s",
                                 ARTDIR,s);
                       13-  59 sprintf(sy,"rm -f %s/%s",dirname,s);

                       13-  66 sprintf(fname,"%s/%s",dirname,s);
           mklist      12-  29 sprintf(a,"%s/%s",dirname,s);
                       12-  33 sprintf(s,"cp %s %s",a,fname);
           outartcls   29-  30 sprintf(fullname,"%s/%s",art_dir,
                                 sub->name);
           output      17-   6 sprintf(t,"/usr/sccsbin/vi %s %s",
                                 filename,outs);



Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     16 /sprintf

           print_ite   18-  25 sprintf(fname,"%s/%s/%s",art_dir,
                                 sub,f);
                       18-  26 sprintf(newfname,"%scomments/%s/%s",
                                 comdir,sub,f);
           publish     20-  28 sprintf(letter,"ltr%d",getpid());
```

```
                20- 29 sprintf(message,"Ohe file %s in
                       SUBJECT %s was accepted                    20- 38 sprintf(
                       );
                20- 55 sprintf(sys,"/usr/we/dock/re
                       cord
                20- 59 sprintf(message,"mv %s %s", fname,
                       tname);
readdirs        21- 45 sprintf(fname,"%s/%s",sub,nf.d_name)
                       ;
readfiles       22- 44 sprintf(fname,"%s/%s",sub,nf.d_name)
                       ;
reject          23- 32 sprintf(letter,"ltr%d",getpid());
                23- 33 sprintf(message,"The file %s in
                       SUBJECT %s was rejected0,
                23- 46 sprintf(mailinglist,"ml%d",getpid())
                       ;
                23- 53 sprintf(sys,"/usr/we/dock/re
                       cord
review          25- 21 sprintf(letter,"ltr%d",getpid());
                25- 22 sprintf(message,"Ohe file %s in
                       SUBJECT %s was accepted                    25- 31 sprintf(
                       );
                25- 48 sprintf(sys,"/usr/we/dock/ar
                       trecord
                25- 52 sprintf(message,"mv %s %s", fname,
                       tname);
                25- 56 sprintf(s,"%scomments/%s/%s/reviewe
                       rs",comdir,sub,f);
                25- 57 sprintf(dirnm,"%scomments/%s",
                       comdir,sub);
                25- 58 sprintf(fcheck,"%s/members", dirnm);

                25- 62 sprintf(message, "mkdirs %s/reviewe
                       rs %scomments/%s/%s",newfname, comdir, sub, f);
                25- 69 sprintf(message,"Please add %s as a
                       review subject",sub);
```

```
Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS      17 /sprintf

          typcheck   27-  15 case 's':        sprintf(s,
                              "%srsubjects/%s",tmpnm,subject);
                     27-  20 case 'r':        sprintf(s,
                              "%scomments/%s/%s/%s",tmpnm

stat      get_subs    7-  33 if(stat(art_dir,&stbuf)  == -1 ||
                      7-  55 if(stat(fullname,&stbuf) == -1)
                              continue;
                      7-  59 stat( fullname, & (subjcts[*subjct_
                              ct]->stbuf));
          main       13-  36 if(stat(dirname,&statinfo)==FALSE)
          mklist     12-  17 if((stat(fname,&statinfo)==FALSE)
                              && (dirname == ' '))
                     12-  30 if(stat(a,&statinfo)==FALSE)
                     12-  46 if(stat(fname,&statinfo)==FALSE)
          publish    20-  39 if(stat(fname,&statinfo)==FALSE)
          readdirs   21-  46 if (nf.d_ino != 0 && stat (fname,
                              &sbuf) >= 0
          readfiles  22-  45 if (nf.d_ino != 0 && stat (fname,
                              &sbuf) >= 0
          reject     23-  42 if(stat(fname,&statinfo)==FALSE)
          review     25-  32 if(stat(fname,&statinfo)==FALSE)
          typcheck   27-  16 stat(s,&statinfo);
                     27-  21 return(stat(s,&statinfo));

strcat    initializ   9-  17 strcat(art_dir,"/");
                      9-  20 strcat(art_dir,"comments");
                      9-  23 strcat(art_dir,types);
                      9-  24 strcat(art_dir,"subjects");
          update     28-  36 strcat (home_subs,"/");
                     28-  37 strcat (home_subs,".user_subs");

strchr    main        1-  46 char *strchr(), *tempt;
                      1-  75 if ((tempt=strchr(opt,argv[i][1]))
                              == 0)
                      1-  77 if ( (tempt=strchr(typ,argv[i][1]))
                              == 0)

strcmp    get_subs    7-  51 if(strcmp(dirbuf.d_name,".") == 0
                              ||
                      7-  52 strcmp(dirbuf.d_name,"..") == 0)
                              continue;
                      7-  82 if(strcmp(subjcts[i]->name,env) ==
                              0)



Tue Jul 17 11:46 1984        X-REF -- FUNCTIONS      18 /strcmp

                      7-  99 if(strcmp(subjcts[i]->name,env) ==
                              0)
          given_sub   8-  27 if(strcmp(subjcts[i]->name,sub) ==
                              0) return(i);
          initializ   9-  19 if( (strcmp(types,"p")==SUCCESS) &&
```

```
                    ioption=='e') )
        submit      26-  31 case 'r':       if(strcmp(s,"rev")
                         == 0 ||
                    26-  32 strcmp(s,"revi")   == 0 ||
                    26-  33 strcmp(s,"revie")  == 0 ||
                    26-  34 strcmp(s,"review") == 0)
                    26-  43 if(strcmp(s,"rej") == 0 ||
                    26-  44 strcmp(s,"reje")   == 0 ||
                    26-  45 strcmp(s,"rejec")  == 0 ||
                    26-  46 strcmp(s,"reject") == 0)
                    26-  52 case 'p':       if(strcmp(s,"p")
                         == 0 ||
                    26-  53 strcmp(s,"pu")     == 0 ||
                    26-  55 strcmp(s,"pub")    == 0 ||
                    26-  56 strcmp(s,"publ")   == 0 ||
                    26-  57 strcmp(s,"publi")  == 0 ||
                    26-  58 strcmp(s,"publis") == 0 ||
                    26-  59 strcmp(s,"publish")== 0)

strcpy  cominfo      3-  10 strcpy(nm,getlogin());
        get_subs     7-  63 strcpy(subjcts[*subjct_ct]->name,
                         dirbuf.d_name);
        initializ    9-  16 strcpy(art_dir,ARTDIR);
                     9-  31 else strcpy(art_subs,env);
                     9-  38 else strcpy(art_opts,env);
                     9-  45 else strcpy(art_excl,env);
        main         1-  60 strcpy(opt,"agocenAGOCEN");
                     1-  61 strcpy(typ,"rsRS");
                     1- 147 if(GOOD) strcpy(sub,sdopt);
        publish     20-  43 strcpy(auth,(getlogin(statinfo.st_u
                         id)));
        reject      23-  47 strcpy(auth,(getlogin(statinfo.st_u
                         id)));
        review      25-  36 strcpy(auth,(getlogin(statinfo.st_u
                         id)));
        submit      26-  36 strcpy(tname,fname);
                    26-  38 strcpy(s,"accepted for review");
                    26-  61 strcpy(tname,fname);
```

```
                    26-  63 strcpy(s,"accepted for publishing");

          typcheck  27-  10 strcpy(nm,getlogin());
          update    28-  35 strcpy (home_subs,homeptr);
                    28-  46 strcpy(usersub.name,subjcts[i]
                              ->name);

strlen    initializ  9-  27 if(env == NULL || strlen(env) == 0)

                     9-  34 if(env == NULL || strlen(env) == 0)

                     9-  41 if(env == NULL || strlen(env) == 0)

          main       1-  71 sprintf(subnames + strlen(subnames),
                              " %s",argv[i]);
          print_ite 18-  23 n=strlen(art_dir) - 9;  /*size of
                              string ssubjects*/
          readdirs  21-  50 while (*p && strncmp (*p, nf.d_name,
                              strlen(*p)))
          typcheck  27-  11 n=strlen(artdir) - 9;   /*size of
                              string ssubjects*/
                    27-  17 n=strlen(nm);

strncmp   main       1-  69 if(strncmp(argv[i],"+",1)!=0)
          mklist    12-  57 if (strncmp(s,"q", 1)== NULL)
                              return;
          print_ite 18-  66 case 'q':      if (strncmp(ptypes,
                              "s",1)== 0) submit(fname,n,sub,f,
          process   19- 156 if (( NOTHERE ) && (strncmp(types,
                              "p",1)!=SUCCESS))
                    19- 168 if (( NOTHERE ) && (strncmp(types,
                              "p",1)==SUCCESS))
          readdirs  21-  50 while (*p && strncmp (*p, nf.d_name,
                              strlen(*p)))
          readfiles 22-  49 while (*p && strncmp (*p, nf.d_name,
                              DIRSIZ))
          typcheck  27-  19 return(strncmp(owner,nm,n));

strncpy   main       1-  84 strncpy(ptypes,tempt,1);
                     1-  92 strncpy(&option,tempt,1);
                     1- 106 strncpy(ptypes,"p",1);
          print_ite 18-  24 strncpy(comdir,art_dir,n);
          readdirs  21-  69 strncpy ((*files)[(*file_ct)-1]
                              .name,
```



```
          readfiles 22-  68 strncpy ((*files)[(*file_ct)-1]
                              .name,
          submit    26-  37 strncpy(&tname[n],"r",1);
                    26-  62 strncpy(&tname[n],"p",1);
          typcheck  27-  12 strncpy(tmpnm,artdir,n);
```

```
strtok     get_subs     7-  28 char *env, *strtok();
                        7-  76 env = strtok(art_subs,":");
                        7-  88 env = strtok(0,":");
                        7-  93 env = strtok(art_subs,":");
                        7- 105 env = strtok(0,":");
           main         1- 133 sdopt = strtok(subnames," ");
                        1- 137 sdopt = strtok(0," ");
                        1- 145 sdopt = strtok(subnames," ");
                        1- 149 sdopt = strtok(0," ");
                        1- 155 sdopt = strtok(0," ");

submit     #DEFN       26-  13 <<DEFN>> submit(fname,n,sub, f,
                                   comdir, nm, newfname)
           print_ite   18-  66 case 'q':       if (strncmp(ptypes,
                                   "s",1)== 0) submit(fname,n,sub,f,

system     cominfo      3-  14 system(sys);
           edit         6-  44 system(s);
           listcreat   10-  44 if (system(sys) == SUCCESS)
           mail        11-   7 system(com);
           main        13-  60 system(sy);
           mklist      12-  34 system(s);
           output      17-   7 system(t);
           publish     20-  56 system(sys);
                       20-  60 system(message);
           reject      23-  54 system(sys);
           review      25-  49 system(sys);
                       25-  53 system(message);
                       25-  63 system(message);

time       main         1-  40 long time();
                        1- 116 subjcts[i]->stbuf.st_mtime = time
                                   ((long *) 0);
           process     19-  10 long time();
                       19- 130 subjcts[substat]->stbuf.st_mtime =
                                   time ((long *) 0);
                       19- 140 subjcts[i]->stbuf.st_mtime = time
                                   ((long *) 0);
```

```
Tue Jul 17 11:46 1984       X-REF -- FUNCTIONS     21 /time

                    19- 192 subjcts[substat]->stbuf.st_mtime =
                                time ((long *) 0);

typcheck  #DEFN     27-   4 <<DEFN>> typcheck(artdir,subject,
                                title,types)
          outartcls 29-  45 (typcheck(art_dir,sub->name,
                                artcls->name,types)==SUCCESS) )

unlink    delete     5-  13 if (unlink(fname)!=SUCCESS)

update    #DEFN     28-  15 <<DEFN>> update (subjcts,subjct_ct)

          main       1- 118 update (subjcts,subjct_ct);
          process   19- 131 update (subjcts,subjct_ct);
                    19- 142 update (subjcts,subjct_ct);
                    19- 193 update (subjcts,subjct_ct);
```

```
Tue Jul 17 11:46 1984       X-REF -- VARIABLES     1  /art_dir

art_dir   #GLOBAL     1-  15 char art_dir[BUFSIZE];
          get_subs    7-  17 get_subs (art_dir,art_subs,art_excl,
                                subjcts,subjct_ct)
                      7-  18 char art_dir[],art_subs[],art_excl[]
                                ;
```

```
                7-  33 if(stat(art_dir,&stbuf)  == -1 ||
                7-  36 fprintf(stderr,"art:
                           name for ARTSUBSO.art_dir);
                7-  40 if((fd = open(art_dir,0)) == -1)
                7-  42 fprintf(stderr,"art: cannot read

                7-  54 sprintf(fullname,"%s/%s",art_dir,
                           dirbuf.d_name);
initializ       9-   7 initialize(art_dir,art_subs,
                           art_opts,art_excl,types,option)
                9-   8 char art_dir[],art_subs[],art_opts[]
                           ,art_excl[],*types,option;
                9-  16 strcpy(art_dir,ARTDIR);
                9-  17 strcat(art_dir,"/");
                9-  20 strcat(art_dir,"comments");
                9-  23 strcat(art_dir,types);
                9-  24 strcat(art_dir,"subjects");
main            1- 108 initialize (art_dir,art_subs,
                           art_opts,art_excl,ptypes,option);
                1- 109 get_subs (art_dir,art_subs,art_excl,
                           subjcts,&subjct_ct);
                1- 115 outartcls(print_item,subjcts[i],
                           art_dir,NEW,ptypes);
                1- 130 process(subjcts,subjct_ct,art_dir,
                           "","",option,ptypes);
                1- 136 process(subjcts,subjct_ct,art_dir,
                           sdopt,"",option,ptypes);
                1- 151 process(subjcts,subjct_ct,art_dir,
                           sub,"",' ',ptypes);
                1- 154 process(subjcts,subjct_ct,art_dir,
                           sub,sdopt,' ',ptypes);
outartcls      29-  18 outartcls (emit,sub,art_dir,flag,
                           types)
               29-  21 char *art_dir,*types;
               29-  30 sprintf(fullname,"%s/%s",art_dir,
                           sub->name);
               29-  45 (typcheck(art_dir,sub->name,
                           artcls->name,types)==SUCCESS) )
```

```
Tue Jul 17 11:46 1984        X-REF -- VARIABLES     2  /art_dir

                      29-  47 (*emit) (artcls->name,sub->name,
                               art_dir,types);
           print_ite  18-   6 print_item (f,sub,art_dir,ptypes)
                      18-   7 char *f, sub[], art_dir[],*ptypes;
                      18-  23 n=strlen(art_dir) - 9:  /*size of
                               string ssubjects*/
                      18-  24 strncpy(comdir,art_dir,n);
                      18-  25 sprintf(fname,"%s/%s/%s",art_dir,
                               sub,f);
           process    19-   4 process(subjcts,subjct_ct,art_dir,
                               sub,item,option,types)
                      19-  30 outartcls(report,subjcts[substat],
                               art_dir,NEW,types);
                      19-  38 outartcls(report,subjcts[i],art_dir,
                               NEW,types);
                      19-  55 outartcls(report,subjcts[substat],
                               art_dir,ALL,types);
                      19-  63 outartcls(report,subjcts[i],art_dir,
                               ALL,types);
                      19-  80 outartcls(report,subjcts[substat],
                               art_dir,EVERYONE,types);
                      19-  87 outartcls(report,subjcts[i],art_dir,
                               EVERYONE,types);
                      19- 104 outartcls(newcount,subjcts[substat],
                               art_dir,NEW,types);
                      19- 111 outartcls(newcount,subjcts[i],
                               art_dir,NEW,types);
                      19- 129 outartcls(print_item,subjcts[substa
                               t],art_dir,ALL,types);
                      19- 139 outartcls(print_item,subjcts[i],
                               art_dir,ALL,types);
                      19- 191 outartcls(print_item,subjcts[substa
                               t],art_dir,NEW,types);
                      19- 199 print_item ( item, sub,art_dir);
                      19- 208 print_item ( item, subjcts[i]->name,
                               art_dir);

art_excl   #GLOBAL     1-  18 char art_excl[BUFSIZE];
           get_subs    7-  17 get_subs (art_dir,art_subs,art_excl,
                               subjcts,subjct_ct)
                       7-  18 char art_dir[],art_subs[],art_excl[]
                               ;
           initializ   9-   7 initialize(art_dir,art_subs,
                               art_opts,art_excl,types,option)



Tue Jul 17 11:46 1984        X-REF -- VARIABLES     3  /art_excl

                       9-   8 char art_dir[],art_subs[],art_opts[]
                               ,art_excl[],*types,option;
                       9-  43 art_excl[0] = ' ';
                       9-  45 else strcpy(art_excl,env);
           main        1- 108 initialize (art_dir,art_subs,
```

```
                                art_opts,art_excl,ptypes,option);
                     1- 109 get_subs (art_dir,art_subs,art_excl,
                                subjcts,&subjct_ct);

art_opts   #GLOBAL      1-  17 char art_opts[BUFSIZE];
           initializ    9-   7 initialize(art_dir,art_subs,
                                art_opts,art_excl,types,option)
                        9-   8 char art_dir[],art_subs[],art_opts[]
                                ,art_excl[],*types,option;
                        9-  36 art_opts[0] = ' ';
                        9-  38 else strcpy(art_opts,env);
           main         1- 108 initialize (art_dir,art_subs,
                                art_opts,art_excl,ptypes,option);

art_subs   #GLOBAL      1-  16 char art_subs[BUFSIZE];
           get_subs     7-  17 get_subs (art_dir,art_subs,art_excl,
                                subjcts,subjct_ct)
                        7-  18 char art_dir[],art_subs[],art_excl[]
                                ;
                        7-  60 if(art_subs[0] == ' ')
                        7-  76 env = strtok(art_subs,":");
                        7-  93 env = strtok(art_subs,":");
           initializ    9-   7 initialize(art_dir,art_subs,
                                art_opts,art_excl,types,option)
                        9-   8 char art_dir[],art_subs[],art_opts[]
                                ,art_excl[],*types,option;
                        9-  29 art_subs[0] = ' ';
                        9-  31 else strcpy(art_subs,env);
           main         1- 108 initialize (art_dir,art_subs,
                                art_opts,art_excl,ptypes,option);
                        1- 109 get_subs (art_dir,art_subs,art_excl,
                                subjcts,&subjct_ct);

ignore     #GLOBAL     22-  20 char *ignore[] = {
                       30-  32 extern char *ignore[];
           readfiles   22-  48 p = ignore;

ignored    #GLOBAL     21-  20 char *ignored[] = {
```

```
Tue Jul 17 11:46 1984        X-REF -- VARIABLES     4  /ignored

          readdirs   21-  49 p = ignored;

interacti print_ite  18-  10 extern int interactive;

ncount    #GLOBAL     1-  25 int ncount;
                     30-  54 extern int ncount;
          newcount   15-   9 extern int ncount;
                     15-  10 if(s) ncount++;
          print_ite  18-  33 extern int ncount;
                     18-  35 ncount++;
          process    19-  26 ncount = 0;
                     19-  31 if(ncount == 0) fprintf(stdout,
                                 "There are no new artcl on
                     19-  51 ncount = 0;
                     19-  56 if(ncount == 0) fprintf(stdout,
                                 "There are no artcl on
                     19-  76 ncount = 0;
                     19-  81 if(ncount == 0) fprintf(stdout,
                                 "There are no artcl on
                     19- 100 ncount = 0;
                     19- 119 if(ncount > 0) fprintf(stdout,
                                 "%d0,ncount);
                     19- 125 ncount = 0;
                     19- 132 if(ncount == 0) fprintf(stdout,
                                 "There are no artcl on
                     19- 186 ncount = 0;
                     19- 194 if(ncount == 0) printf("There are
                                 no new artcl on
                     19- 200 if(ncount == 0) printf("
                                 not a artcle on
                     19- 210 if(ncount == 0) printf("There is no
                                 artcle called
          report     24-  12 extern ncount;
                     24-  25 ncount = 5;
                     24-  27 if(ncount%5 == 0) printf("      ");

                     24-  29 ncount++;

save_addr #GLOBAL     1-  22 jmp_buf save_addr;
                     30-  51 extern jmp_buf  save_addr;
          onintr     16-   8 longjmp(save_addr, 1);
          print_ite  18-  42 if (setjmp(save_addr))



Tue Jul 17 11:46 1984        X-REF -- VARIABLES     5  /stdbuf

stdbuf    #GLOBAL     1-  14 char stdbuf[BUFSIZ];
                     30-  49 extern char stdbuf[BUFSIZ];

subnames  #GLOBAL     1-  19 char subnames[BUFSIZE];
          main        1-  71 sprintf(subnames + strlen(subnames),
```

```
          " %s",argv[i]);
1- 110 if ( subnames[0] == ' ' && option
          == EMPTY)
1- 129 if(subnames[0] == ' ')
1- 133 sdopt = strtok(subnames," ");
1- 145 sdopt = strtok(subnames," ");
```

```
Tue Jul 17 11:46 1984     X-REF -- DEFINITIONS     1  /ALL

ALL                art.h  30-  14 #define ALL              1
ARTDIR       initialize.c   9-   1 #define ARTDIR "/usrb/we/dock
A_BOARD            art.h  30-  19 #define A_BOARD          *(argv
BAD                art.h  30-  1C #define BAD              (subst
ARTDIR         mklistcom.c  13-   2 #define ARTDIR "/usrb/we/dock
BOARD              art.h  30-  18 #define BOARD            *argv
BUFSIZE            art.h  30-  24 #define BUFSIZE 256
EMPTY              art.c   1-  12 #define EMPTY ''
EVERYONE           art.h  30-  13 #define EVERYONE         2
EXIST              art.h  30-  16 #define EXIST            0
FALSE              art.h  30-  27 #define FALSE -1
GOOD               art.h  30-  12 #define GOOD             (subst
ILL                art.c   1-   3 #define ILL "   illegal opti
ILLTOG             art.c   1-   4 #define ILLTOG "   illegal o
INDENT             art.h  30-  23 #define INDENT           3
MAXBDS             art.h  30-  25 #define MAXBDS 20
NEW                art.h  30-  15 #define NEW              0
NO                 art.c   1-  11 #define NO 0
NOTHERE            art.h  30-  11 #define NOTHERE          (subst
OPTION             art.h  30-  21 #define OPTION           (*argv
SAME               art.c   1-   9 #define SAME 0
SUCCESS            art.h  30-  17 #define SUCCESS          0
TITLE              art.h  30-  20 #define TITLE            *(argv
TRUE               art.h  30-  26 #define TRUE 1
USE                art.c   1-   2 #define USE "art [+[agocen]] [
YES                art.c   1-  10 #define YES 1
subjcts            art.h  30-  40 struct subjcts
usersub            art.h  30-  34 struct usersub
```

Designing and Implementing a Computer Conferencing System to Manage and Track Articles Through the Revision Process

by

Patricia Dock

B. A. University of West Florida, 1979

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

1984

Designing and Implementing a Computer Conferencing System to Manage and Track Articles Through the Revision Process

by Patricia A. Dock

An Abstract of a Master's Report

This paper presents ARTHER (ARTicle HandlER), a conferencing system, designed to provide a mechanism for tracking articles through the various stages of revision. The system was implemented on a UNIX based system. The design, implementation, and possible enhancements are discussed. A sample session using ARTHER is contained in the report. Appendices contain a copy of the code as well as the manual pages for the implementation.

The services provided by ARTHER are simple and easy to use by a community of users. It is intended to interface with and compliment the existing services on a UNIX based system. Articles are smallest element in the system. They are grouped in subjects. The owner of the subject controls the articles from the time of submission to the system through the time that they are deemed acceptable for publication by the owner of the subject. During this process, the owner of the subject has the opportunity to name a list of "experts" to act as reviewers for the article. These reviewers have an opportunity to comment on the article. Access to the article and reviewers's comments are controlled.