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:

[Signature]

Major Professor
1. Introduction .......................................................... 1
  1.1 Overview ...................................................... 1
  1.2 High Level Description ...................................... 2
    1.2.1 Original Author ......................................... 3
    1.2.2 Owner of the Subject ................................... 3
    1.2.3 Members of the Subject ................................ 3
    1.2.4 Reviewers ................................................. 3
    1.2.5 Mail .......................................................... 4
    1.2.6 Informed ................................................... 4
    1.2.7 Submitted ................................................ 4
    1.2.8 Status of the Article ................................... 4
    1.2.9 Accepted for Publication .............................. 4
    1.2.10 Accepted for Review .................................. 4
    1.2.11 Rejected .................................................. 4
    1.2.12 More on the Status of the Article ................. 5
    1.2.13 Comments ................................................ 6
  1.3 Design Assumptions ......................................... 6

2. Description of ARThER .............................................. 7
  2.1 Range of Services ........................................... 7
  2.2 Getting Started .............................................. 8
  2.3 Tracking ..................................................... 9
  2.4 Types of Input ............................................... 10
  2.5 System in Use ............................................... 11

3. Justification and Comparisons .................................. 22
  3.1 Justification ................................................ 22
  3.2 Comparisons and Contrasts ................................ 25
    3.2.1 VMSHARE .................................................. 25
    3.2.2 SCOOP ..................................................... 26
    3.2.3 TELECENTER ............................................... 29
    3.2.4 SCCS ....................................................... 30

4. Summary and Future Enhancements .............................. 32

BIBLIOGRAPHY ...................................................... 34
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.
THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF
LIST OF FIGURES

Figure 1. System Configuration - Initial state .................. 12
Figure 2. Creating a Subject ........................................ 13
Figure 3. Submitting an Article ...................................... 14
Figure 4. System Configuration - artssubmit ...................... 15
Figure 5. Notification of Submitted Article ....................... 16
Figure 6. Viewing a Submitted Article ................................ 17
Figure 7. Entering Comments ......................................... 18
Figure 8. Viewing a Submitted Article continued ............... 19
Figure 9. System Configuration - art +s ........................... 20
Figure 10. Notification of Articles for Review .................... 22
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 guidance 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 similar services. Included in these systems are VMSHARE, TELECENTRE, 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 and intended to run on a UNIX 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 setttable 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 ARATHER 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 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 be 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

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 .ARTHET in the user's home directory. It then allows the user's to select their own values for command options. Finally it modifies the user's .profile in the HOME directory to initialize .ARTHET 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 ARATHER.

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.
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

arsubmit
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 "arsubmit" was completed.
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: test! 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.
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, one 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 ARTHR, 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.

ARTHHER 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 ARTHR 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 ARTHR.

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.

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.

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 ARATHER, 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 ARATHER), and the title of the newspaper article. To submit an article to ARATHER, 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 submitted to ARThER is the stimuli to the subject owner to enter a list of reviewers.

- If no referee is supplied 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 similar 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 arts submitting command creates a directory structure for the file in an existing directory structure which was created when the arts subject 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".

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

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.

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

- Unlike ARHER, it was designed for an environment in which one wishes to keep the "old" versions available. In ARHER, 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 ARHER. The purpose of SCCS is to have the ability to recreate all versions of the module at any time. The intent of ARHER 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 ARHER 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" has access to the system. If the UNIX command "uucp" 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.

BIBLIOGRAPHY


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

artssubject

SYNOPSIS

artssubject

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

/ARTHERRORDIR/ssubjects directory is created.

/ARTHERRORDIR/rsubjects directory is created.

/ARTHERRORDIR/psubject directory is created.

SEE ALSO

artssubmit art artprofiler
NAME

artsubmit

SYNOPSIS

artsubmit

DESCRIPTION

The artsubmit command accepts a file to be copied into 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 ARATHER 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
/SHOME/.profile /SHOME/.artprofile

SEE ALSO
art artsubject artssubmit 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 artscommit 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/ARTHERR/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/ARTHERR 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/ARTHERR", as long as the
user's current directory is one level lower than the
directory which is to be used as the ARTHERR system
directory.

SEE ALSO
  make replace overwrite
NAME
   replace

SYNOPSIS
   replace

DESCRIPTION
   The replace command replaces all occurrences 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
# define USE "art [+agocen] [+rs] [subname] [title]0
#define ILL "illegal option %s usage: %s"
#define ILLTOG "illegal option %s and %s 
  - togetherUsage: %s"
#include "art.h"

/* struct for command and env variable parsing */
define SAME 0
define YES 1
define NO 0
define EMPTY ""

char stdout[BUFSIZE];
char art_dir[BUFSIZE];
char art_subs[BUFSIZE];
char art_opts[BUFSIZE];
char art_excl[BUFSIZE];
char subnames[BUFSIZE];

jmp_buf save_addr;

int ncount;
long time();

extern char *strtok();

main (argc,argv)
int argc;
char *argv[];
(
  struct subjects *subjects[MAXBDS];
  int subject_ct, substat, i;
  char sub[BUFSIZE];
  int print_item();
  char option;
  long time();

  char types,*ptypes;
char opt[13];
char typ[7];
char *strchr(), *tempt;
char *sadopt;
int runcmd;

/*
 * initialization
*/

runcmd = YES;
option = EMPTY;
types = EMPTY;
ptypes= &types;
strcpy(opt, "agocenAGOCEN");
strcpy(typ, "rsRS");

i=1;

while (argc -- >1)
{
    if (strcmp(argv[1], "++", 1)!=0)
    {
        sprintf(subnames + strlen(subnames), "%s", argv[1]);
    }
    else
    {
        if (((tempt=strchr(opt, argv[1][1])) == 0)

        {
            if (((tempt=strchr(typ, argv[1][1])) == 0)

            {
                printf(ILL.argv[1], USE);
                runcmd=NO;
                continue;
            }
If (types==EMPTY) {
    strncpy(ptypes.temp,1);
    else {
        printf(ILLTG . ptypes . temp . USE);
        runcmd=NO;
    }
}
else if (option==EMPTY) {
    strncpy(&option.temp,1);
    else {
        printf(ILLTG . ptypes . temp . USE);
        runcmd=NO;
    }
}
++;
}
if (runcmd == NO) exit(1);
if (types==EMPTY) {
    strncpy(ptypes,"p",1);
    /*more initialization*/
    initialize (art_dir,art_sub,art_opt,art_excl, ptypes,option);
    get_subs (art_dir,art_sub,art_excl,subjcts, &subjct_ct);
}
if (subnames[0] == '/' && option == EMPTY) {
    for (i=0 ; i < subjct_ct ; i++) {
        if(subjcts[i]->excl == 1) continue;
        outputcl (print_item,subjcts[i],
                  art_dir,NEW,ptypes);
        subjcts[i]->stbuf.st_mtime = time ((long *) 0);
    }
    update (subjcts,subjct_ct);
    exit();
}
/* process arguments */
/* check for options, process command */

if (option != EMPTY)
{
    if (subnames[0] == ' ')
        process(subjects.subject_ct.art_dir,
            """".option,ptypes);
    else
    {
        sdopt = strtok(subnames," ");
        while (sdopt != NULL)
        {
            process(subjects.subject_ct,
                art_dir,sdopt,"".option,ptypes);
            sdopt = strtok(0," ");
        }
    }
}

/* no option, just sub and/or news items */
else
{
    sdopt = strtok(subnames," ");
    substat = given_sub(sdopt,subjects.subject_ct);
    if (GOOD) strcpy(sub,sdopt);
    else sub[0] = ' ';
    sdopt = strtok(0," ");
    if (GOOD && sdopt != NULL)
    {
        process(subjects.subject_ct,art_dir,
            sub,"""",ptypes);
        while (sdopt != NULL)
        {
            process(subjects.subject_ct,art_dir,
                sub,sdopt,"","ptypes);
            sdopt = strtok(0," ");
        }
    }
}
/*
convert arbitrary string to standard file name
*/

#include <stdio.h>
#include <ctype.h>

main(argc,argv)
int argc:
char **argv;
{
    int i,j,k:
    k = 0:
    for(i = 1; i < argc; i++)
    {
        j = 0:
        while( argv[i][j] != '"')
        {
            if(isalnum(argv[i][j]) ||
               argv[i][j] == '+' ||
               argv[i][j] == '-' ||
               argv[i][j] == '-' ||
               argv[i][j] == '_')
            {
                puts(argv[i][j],stdout):
                if( ++k == 14) exit(0):
            }
            j++:
            if(++k == 14) break:
            if(i+1 == argc) break:
            puts('"',stdout):
        }
        puts(0,stdout):
    }
# include "art.h"

cominfo(dirnm,nm)

char *dirnm, *nm;
{
  char *getlogin(), s[BUFSIZE], sys[BUFSIZE]:

  strcpy(nm,getlogin());
  sprintf(s,"%s/%s",dirnm,nm);
  chdir(s):
  sprintf(sys,"/usr/we/dock/artrecord
    reviewed article for publication
    system(sys):
    printf(" 
      the article.");
    printf(" You will be unable to write the 
      file.");
    printf(" To enter comments for the author 
      to read,");
    printf(" simply type 
      can enter");
    printf(" comments. REMEMBER: the author will 
      know what");
    printf(" article you are commenting upon. but 
      not where");
    printf(" you are in the program. To exit the 
      article");
    printf(" simply type 
      Type a carriage return when ready 
      for the file.");
  gets(s):
}

/*
   artcomment.c
   Assumes that one is in the directory where
   one wishes the comments to be written in a
   file named comment.
   If called from artI+us then the directory
   should be  artdir/comments/$SUBJECT/$TITLE/$REVIEWER
   for the articles which are being reviewed.
   The directory should be
   artdir/comments/$SUBJECT/$TITLE
   for files which are considered "published."
*/

#include <stdio.h>

main()
{
    char s[80];
    char NEWLINE;
    FILE *fp;
    s[0]=';
    fp=fopen("comment"."a");
    printf(" Enter your comments a line at a time. ");
    printf(" When you have completed your comment. ");
    printf(" Begin a line with a period (. )");
    printf(" followed by cariage return. ");
    printf(" ");
    for (gets(s); s[0] != '.' && s[1] != ' ';){
        fputs(s,fp);
        fputs('0',fp);
        gets(s);
    }
    fclose(fp);
}
1:

2:

3: /*
4:   delete.c
5:   removes a file named 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:
/* edits a list, giving an opportunity to keep or remove each entry in the list.
called edit(fname)
*/

#include "art.h"

edit(fname)

char *fname;

(char *tmpfile[BUFSIZE], char ans[10], char s[BUFSIZE], FILE *fopen(), *tmp, *fp;

int getpid();

s = sprintf(tmpfile, "tmp%d", getpid());

if((tmp=fopen(tmpfile,"w"))!= NULL)
    printf("Unable to open %s", tmpfile);

if((fp=fopen(fname,"r"))!= NULL)
    printf("Unable to open %s", fname);

while(fgets(s, sizeof s, fp)!=NULL)
{
    printf("9s: keep or delete [keep]?", s);
    gets(ans);
    switch(ans[0])
    {
    case '?': printf(" keep: entry remains in list");
                printf(" delete: removes entry from list");
    case 'k': puts(s,tmp);
                puts("0.tmp");
    }
}
41:     fclose(fp);
42:     fclose(tmp);
43:     sprintf(s,"mv %s %s.tmpfile, fname);
44:     system(s);
45: }
46: }
47: 
48: 
49: 
50: 
51: 
/*
get_subs

read names and last modified times of subs

arguments

allsubs and ptrs of allocated memory
usersub where files are read in
allsub_ct and number of subs found
usersub_ct in each file

enough memory is allocated for usersub to fit all possible subs
*/

#include "art.h"

get_subs (art_dir.art_subs.art_excl.subjects.subject_ct)
char art_dir[].art_subs[].art_excl[];
struct subjects *subjects[MAXBDS];
int *subject_ct;
{
struct usersub usersub;
struct stat stbuf;
struct direct dirbuf;
int i,fd;
char *malloc();
FILE * fopen(), *infie;
char * env, * strok();
char fullsize[BUFSIZE];

/* get sub names */

if(stat(art_dir,&stbuf) == -1 ||
   (stbuf.st_mode & S_IFMT) != S_IFDIR)
{
   fprintf(stderr,"art:
       ARTSUBSO.art_dir);
   exit(1);
}

if((fd = open(art_dir,0)) == -1)
{

```c
42:         fprintf(stderr,"art: cannot read
43:             sub0, art_dir");
44:         exit(1);
45:     }
46:     /* read in file names */
47:     *subject_ct = 0;
48:     while(read(fd,(char *)&dbuf,sizeof(dbuf))>0)
49:     {
50:         if(dbuf.d_ino == 0) continue;
51:         if(strcmp(dbuf.d_name, ".") == 0 ||
52:             strcmp(dbuf.d_name, "..") == 0)
53:             continue;
54:             /* check for directory, modified times */
55:             sprintf(fullname, "%s/%s", art_dir,
56:                     dbuf.d_name);
57:             if(stat(fullname, &dbuf) == -1) continue;
58:             if((dbuf.st_mode & S_IFMT) != S_IFDIR)
59:                 continue;
60:             /* add information to subjects */
61:             subjects[*subject_ct] = malloc(sizeof (struct subjects));
62:             stat(fullname, & (subjects[*subject_ct]->dbuf));
63:             if(!art_subs[0] == '')
64:                 subjects[*subject_ct]->excl = 0;
65:             else subjects[*subject_ct]->excl = 1;
66:             strcpy(subjects[*subject_ct]->name,
67:                     dbuf.d_name);
68:             subjects[(*subject_ct++)->new = 1;]
69:     }
70:     /* check for new subs */
71:     for(i = 0; i < *subject_ct; i++)
72:     {
73:         if(subjects[i]->new == 0) continue;
74:         subjects[i]->dbuf.st_mtime = OL;
75:     }
76:     /* check for valid subs */
77:     env = strtok(art_subs, ":");
78:     if (env[0] == '+') env++;     /* skip leading */
```
while(env != NULL)
{
    for(i = 0; i < *subject_ct; i++)
    {
        if(strcmp(subjects[i]->name.env) == 0)
        {
            subjects[i]->excl = 0;
            break;
        }
    }
    env = strtok(0,".:");
    if (env[0] == '"') env++; /* skip leading */
}
/* check for excluded subs */
env = strtok(aot,subs,".");
if (env[0] == '"') env++; /* skip leading */
while(env != NULL)
{
    for(i = 0; i < *subject_ct; i++)
    {
        if(strcmp(subjects[i]->name.env) == 0)
        {
            subjects[i]->excl = 1;
            break;
        }
    }
    env = strtok(0,".:")
    if (env[0] == '"') env++; /* skip leading */
}
given_sub
-1 if a nonexistent sub is passed
0 if NULL is passed
1 if sub is a member of sublist
and i = its index into the list

#include "art.h"
given_sub (sub.subjects.subject_ct)
struct subjects *subjects[MAXBDS];
int subject_ct;
char *sub;
{
    int i;
    if ( sub == NULL || *sub == ' ')
        return (-1);
    /* linear search thru sublist for sub */
    for(i = 0; i < subject_ct; i++)
    {
        if(strcmp(subjects[i]->name, sub) == 0)
            return(i);
    }
    return (subject_ct);
#define ARTDIR "/usr/we/dock"
#include "art.h"
#include <stdio.h>
#include <signal.h>

initialize(string dir, string subs, string opts, string excl, types, option)
char art_dir[], art_sub[], art_opt[], art_exc[], "types.
    option:
{
    extern _exit();
    char *env,*getenv();
    if (signal(SIGQUIT, SIG_IGN) != SIG_IGN)
        signal(SIGQUIT, _exit);

    // get environment variables */
    strcpy(art_dir, ARTDIR);
    strcat(art_dir,"/");

    if( (strcmp(types,"p") == SUCCESS) && (option == 'e') )
    strcat(art_dir,"comments");
    else
    {
        strcat(art_dir,types);
        strcat(art_dir,"subjects");
    }

    env = getenv("ARTSUBS");
    if(env == NULL || strlen(env) == 0)
    {
        art_sub[0] = ' ';;
    }
    else strcpy(art_sub, env);

    }
/* creates a list of name fname.
looks in file fcheck for validation of names
called creat(fname,fcheck)
*/

#include "art.h"

listcreate(fname,fcheck)
char *fname, *fcheck;
{
FILE *fp;
char s[BUFSIZE];
char sys[BUFSIZE];
int answer;

if ((fp=fopen(fname,"a")) == NULL)
    printf(" can't open fname");

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

printf(" name: ");
gets(s);

while (answer != TRUE)
    switch (s[0])
    {
    case '
: printf(" Enter names one at a time");
    printf(" when finished, type a period");
    printf(" followed by a carriage return.");
    printf(" name: ");
    gets(s);
    break;
    case '.':
        if (s[1] == NULL)
            answer = TRUE;
        break;
    }
default:
    break;
    
    if (system(sys) == SUCCESS)
        puts(s.fp);
        puts("0,fp");
    
else
    printf(" invalid
        entry");
    
printf(" name: ");
    gets(s);
    
    close(fp):
    
}
1: mail(letter, mailinglist)
2: char *letter, *mailinglist;
3: {
4: char com[256];
5: sprintf(com, "/usr/we/dock/ml %s %s", mailinglist, letter);
6: system(com);
7: }
#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 exist. Do you wish to duplicate an existing list? ");
            printf("List for this subject? ");
            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;
            printf(s, "cp %s
%s", a, fname);
            system(s);
            case 'n':
                answer=TRUE;
case '?': break;
}

/* If we are here, we have already copied the file
   if one was wanted. If fname does not exist,
   create it empty and prompt for names to add to
   it. */

if(stat(fname,&statinfo)==FALSE)
{
    listcreate(fname,fcheck);
}

/* Give user chance to edit the file */

printf(" Do you wish to edit the file or quit? ");
gets(s);
if (strcmp(s,"q",1)==NULL) return;
edit(fname);
}
#include "art.h"
define ARTDIR "/usr/we/dock"

int whoami;
int answer;
char dirname[BUFSIZE];
char fname[BUFSIZE];
char s[BUFSIZE];
char sy[BUFSIZE];
struct stat statinfo;

/* Get the uid of the person executing command*/
whoami=geteuid();

printf("Lists are created on a subject basis by the owner of the group.");

for (answer=0; answer!=TRUE;)
{
    printf(" subject: ");
s[0] = ' ';
    if (gets(s) == NULL) exit(1);
    switch(s[0])
    {
        case '?':
            printf("Enter the name of the subject for which you wish to");
            printf(" create the list. You must be the owner of the group");
            printf(" Type quit to exit.");
            break;
        case NULL:
            default:
            printf(dirname,"%s/comments/%s",ARTDIR,s);
            if(stat(dirname,&statinfo)==FALSE)
            {
                printf(" %s is not a valid subject",s);
                break;
            }
    }
}
40:     }
41:     if (statinfo.st_uid != whoami)
42:     {
43:         answer=TRUE;
44:     }
45:     else
46:         printf(" You are not the owner of
47:                 the group %s",s);
48:     }
49: }
50:     printf(" Do you wish to create a new list, delete
51:            an existing list.");
52:     printf(" or edit an existing list?");
53:     printf(" create, delete, or edit? ");
54:     gets(s);
55:     switch (s[0])
56:     {
57:         case 'd':
58:             printf(" name of existing list: ");
59:             gets(s);
60:             sprintf(sy,"rm -f %s/%s",dirname,s);
61:             system(sy);
62:             exit(0);
63:         default:
64:             printf(" Name of the list you wish to
65:                    create or edit? ");
66:             gets(s);
67:             sprintf(fname,"%s/%s",dirname,s);
68:             mklist(fname,dirname, "/etc/passwd");
69:     }
#include "art.h"

int main(argc, argv)
    int argc;
    char *argv[];
{
    char s[BUFSIZE];
    listcreate(argv[1], "/etc/passwd");
    for (answer=0; answer!=TRUE;)
    {
        printf("Do you wish to edit the file or quit? ");
        gets(s);
        switch (s[0])
        {
            case 'e': edit(argv[1]);
                answer=TRUE;
                break;
            case 'q': exit(1);
            case '?': printf("Edit: examine the entries one at a time");
                printf("Quit: leave the list as it is.");
                edit(argv[1]);
        }
    }
}
/*
 * newcount
 * counts number of artcl
 */

newcount (s, sub)
    char *s,
    sub[];

( extern int ncount;
  if(s) ncount++;
1: *
2: #include "art.h"
3: 
4: onintr()
5: {
6:     fprintf(stderr,"0interrupt0");
7:     longjmp(save_addr, 1);
8: }

1: output(filename, outs)
2: char filename[], outs[];
3: {
4: char t[256];
5: 
6:  sprintf(t,"/usr/ccs/bin/vi %s %s", filename, outs);
7:  system(t);
8: }
```c
#include "art.h"

FILE *fd;
extern int interactive;
char s[BUFSIZE];
int n, i;
char fname[BUFSIZE];
char newfname[BUFSIZE];
char nm[10];
char cmdir[BUFSIZE];
static int firstitem = 1;
int onintr();

if (f == NULL) {
    return;
}

n=strcmp(art_dir) - 9; /*size of string ssubject*/
strncpy(cmdir,art_dir,n);
sprintf(fname,"%s/%s/%s",art_dir,sub,f);
sprintf(newfname,"%scomments/%s/%s",cmdir,sub,f);

if ((fd = fopen (fname, "r")) != NULL)
{
    register int c, ip, op;
    struct stat sbuf;
    char *ctime();
    extern int ncount;

    ncount++;

    fstat (fileno (fd), &sbuf);
    if (firstitem) {
        firstitem = 0;
        putchar (40);
    }
    if (setjmp(save_addr))
        goto finish;
```
if (signal(SIGINT, SIG_IGN) != SIG_IGN)
    signal(SIGINT, onintr);

s[0] = ' ';
printf(" Subject: %s article:
 %s0.sub.f);
cominfo(newfname,nm);
output(fname,&s[0]);

finish:
    putchar('0');
    fclose(fd);
    if (signal(SIGINT, SIG_IGN) != SIG_IGN)
        signal(SIGINT, SIG_DFL);

prompt:
s[0] = ' ';
printf(stderr," review, next, quit ? [next"
    " ]
    gets(s);
    switch(s[0])
    {
    case 'r':
        putc('0',stdout);
        goto start;
        break;
    case 'n':
    case 'q':
        if (strncpy(potypes,"s",1) ==
            0) submit(fname,n,sub,f,comdir,nm,newfname);
        exit(1);
        break;
    case '?':
        fprintf(stderr," %10s%0."review,"print article again"");
        fprintf(stderr," %10s%0."next","go to next article"");
        fprintf(stderr," %10s%0."quit","terminate program"");
        goto prompt;
        break;
    case ' ': break;
    default: fprintf(stderr,"unknown
 command, please try again
 0);
        goto prompt;
break;
}
putc('0', stdout);
}
#include "art.h"

process(subjcts, subjct_ct, art_dir, subjct, option, types)
struct subjcts *subjcts[MAXBDS];
int subjct_ct;
cnr *subjct, option, *types;

int newsat, stat, j;
long time();
int print_item(), report(), count(), newcount();
int flag;

/* check option for all subs */
if ( isupper(option) )
{
    flag = 2;
}
elser flag = 1:

switch (option )
{
    case 'n':    /* report titles of new artcl */
    case 'N':
        ncount = 0;
        stat = given_sub(subjcts, subjct_ct);
        if ( GOOD )
            {
                outartcls(report, subjcts[stat],
                    art_dir, NEW, types);
                if (ncount == 0) printf(stdout,
                    "There are no new artcl on
                }else if ( NOTHERE )
        {
            for (i=0 ; i < subjct_ct ; i++)
            {
                if(subjcts[i]->excl == flag)
                    continue;
                outartcls(report, subjcts[i],
                        art_dir, NEW, types);
            }
else if ( BAD )
{
    fprintf(stderr, "existing sub.C.sub):
    }
}
break:

    case 'O':    /* report appropriate titles of */
        /* articles regardless of currency */
    case 'O':
    ncount = 0;
    substrat = given_sub[subject_count.ct]:
    if ( GOOD )
{
    outatcls(report.subjects[substrat],
        art_dir.ALL.types);
    if(ncount == 0) fprintf(stdout,
        "There are no artcls on
    }
else if ( NOTHERE )
{
    for (i=0; i < subject_count; i++)
    {
        if(subjects[i]->excl == flag) continue;
        outatcls(report.subjects[i],
            art_dir.ALL.types):
    }
    }
else if ( BAD )
{
    fprintf(stderr,"existing sub.C.sub):
    }
}
break:

    case 'e':    /* report all titles of articles */
    case 'E':
ncount = 0;
substat = given_sub(sub.subjcts,subjct_ct);
if ( GOOD )
{
    outartcls(report.subjcts[substat],
              art_dir,EVERYONE.types);
    if(ncount == 0) fprintf(stdout,
         "There are no artcls on
    
else if ( NOTHERE )
{
    for (i=0 ; i < subjct_ct ; i++)
    {
        outartcls(report.subjcts[i],
                   art_dir,EVERYONE.types);
    }
}
else if ( BAD )
{
    fprintf(stderr," existing sub:0.sub);
}
break;
case 'c': /* count new artcls */
case 'C':
ncount = 0;
substat = given_sub(sub.subjcts,subjct_ct);
if ( GOOD )
{
    outartcls(newcount,subjcts[substat],
              art_dir,NEW.types);
}
else if ( NOTHERE )
{
    for (i=0 ; i < subjct_ct ; i++)
    {
        if(subjcts[i]->_excl == flag)
            continue;
        outartcls(newcount,
                   subjcts[i],art_dir,NEW.types);
else if ( BAD )
{
    fprintf(stderr, "existing sub.C.sub JJ

if ( ncount > 0 ) fprintf(stdout, "%d. ncount"
;
break;

case 'a': /* print contents of all artcls regardless of currency */
case 'A':
    ncount = 0;
    substat = given_sub(sub.subjects, subject Ct);
    if ( GOOD )
    {
        outartcls(print_item,
            subjects[subject].art_dir.ALL.types);
        subjects[subject].art.dir. ALL.types->stbuf.st_mtime =
            time ((long *)0);
        update (subjects, subject Ct);
        if ( ncount == 0 ) fprintf(stdout,
            "There are no artcls on

        else if ( NOTHERE )
        {
            for (i=0 ; i < subject Ct ; i++)
            {
                if ( subjects[i].art_dir. ALL.types->stbuf.st_mtime
                    = time ((long *)0);
                update (subjects, subject Ct);
            }
        }
        else if ( BAD )
    {
fprintf(stderr, "existing sub.0.sub:

break;

case 'g': /* report all sub names */
case 'G':
    substat = given_sub(sub.subjects, subject_ct);
    if (GOOD)
        fprintf(stdout, "sub:
    if ((NOTHERE) && (strcmp(types["p", .1] !=SUCCESS))
    
    fprintf(stdout, "existing subjects
        for (i=0 ; i < subject_ct ; i++)
        {
            if(subjects[i]->excl == flag)
                continue;
                if((j++)%5 == 0) printf(
            "
        );
        fprintf(stdout, "%15.14s", subjects[i]->name);
    }
    putc('0', stdout);
    }
    if ((NOTHERE) && (strcmp(types["p", .1] ==SUCCESS))
    
    fprintf(stdout, "existing subjects:
        for (i=0 ; i < subject_ct ; i++)
        {
            if((j++)%5 == 0) printf(""
        );
        fprintf(stdout, "%15.14s", subjects[i]->name);
    }
    putc('0', stdout);
178:     
179:         if ( BAD )
180:             fprintf(stdout,"
181:                  O.sub);
182:             break;
183:       
184:       case 't':
185:           substat = given_sub(sub.subjects.subject_ct);
186:           newstat = given_sub(item.subjects.subject_ct);
187:           ncount = 0;
188:       
189:           /* if a sub, print artcl there */
190:           if(GOOD &amp; newstat == -1)
191:             {
192:               outartcls(print_item,
193:                         subct[subjstat].art_dir,NEW.types);
194:               subct[subjstat]-&gt;stbuf.st_mtime =
195:                         time ((long *) 0);
196:               update (subct,subject_ct);
197:               if(ncount == 0) printf("There are
198:                         no new artcl on
199:               }
200:                 /* if sub &amp; title, print it */
201:             else if ( GOOD)
202:                 {
203:                 print_item ( item, sub,art_dir);
204:                 if(ncount == 0) printf("n
205:             not a artcle on
206:             }
207:             /* if just title, print all such titles */
208:             else
209:             {
210:                  for( i = 0; i &lt; subject_ct ; i++)
211:                     {
212:                         if(subjects[i]-&gt;excl == flag)
213:                             continue;
214:                         print_item ( item,
215:                                 subjects[i]-&gt;name,art_dir);
216:                         if(ncount == 0) printf("There is no
217:                                 artcle called
218:                         }
212:        break;
213: 
214:    default:
215:        fprintf(stderr, "art:
216:            option:
217:                exit (1);
218:        break;
219:    }

#include "art.h"

/*
 main()
{
  char fname[BUFSIZE], sub[BUFSIZE],
      f[BUFSIZE], comdir[BUFSIZE], tname[BUFSIZE];
  int n;
  strcpy(fname,
          "/usr/ww/dock/ssubjects/test1/file1");
  strcpy(sub, "test1");
  strcpy(f, "file1");
  strcpy(comdir,
          "/usr/ww/dock/comments/test1/file1");
  strcpy(tname,
          "/usr/ww/dock/psubject/test1/file1");
  publish(fname, n, sub, f, comdir, tname);
}
*/

publish(fname, n, 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, "ltn%d", getpid());
sprintf(message, "One file %s in SUBJECT %s was accepted 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:       scanf(mailinglist,"msg%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:               /* mail the info */
48:               mail(letter, mailinglist);
49:               delete(letter);
50:               delete(mailinglist);
51:               /* record for posterity */
52:               printf(sys,"/usr/ker/record
53:               " accepted for publication
54:               system(sys);
55:               /* move the file to new location */
56:               printf(message,"mv %s %s", fname, tname);
57:               system(message);
58:               }
59:       }
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: #include "art.h"
18:
19: char *ignored[] = {
20: ".
21: ",
22: ".
23: NULL
24: };
25:
26: readdirs (sub, file_ct, files)
27: char *sub;
28: int *file_ct;
29: struct user *sub **files;
30: {
31: struct direct nf;
32: struct stat sbuf;
33: char fname[BUFFSIZE];
34: FILE *fd;
35: int i, j;
36: char *malloc(), *realloc();
37: if ((fd = fopen (sub, "r")) == NULL) {
38: return (1);
39: }
40: } /* Read the file names into files */
*file_ct = 0;
while (fread ((char *) &nf, sizeof nf, 1, fd) == 1)
{
    snprintf (fname, "%s/%s", sub, nf.d_name);
    if (nf.d_ino == 0 && stat (fname, &sbuf) == 0
        && ((sbuf.st_mode & S_IFDIR) == S_IFDIR)) {
        register char **p;
        p = ignored;
        while (*p && strncmp (*p, nf.d_name, strlen(*p)))
            ++p;
        if (!*p) {
            if (*file_ct++ > 0)
                *files = (struct
                    usersub *)
                realloc ((char *) *files,
                    (unsigned)
                    (sizeof
                        (struct usersub)
                        * (*file
                         e_ct)));
            else
                *files = (struct
                    usersub *) malloc
                ((unsigned)
                    (sizeof
                        (struct usersub)
                        * (*file
                         e_ct)));
            if (*files == NULL) {
                fprintf (stderr,
                    "No storage!");
                exit (1);
            }
        }(*files)[(*file_ct)-1]
            .mtime = sbuf.st_mtime;
        strncpy (*files)[(*file_ct)
            -1].name,
            .nf.d_name, DIRSIZ);
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: /* Clean up */
85: fclose (fd);
86: return (0);
readfiles.c 22-1 /1

/*
 * readfiles
 * reads all regular file names and modification times
 * in the present working directory
 * allocates space as each entry is assigned to the
 * structure
 * sorts entries in decreasing time order
 * arguments
 * file_ct    number of files found
 * files     structure to fill
 * returns
 * 1 if cannot open "."
 * 0 otherwise
 */

#include "art.h"

char *ignore[] = {
    "core",
    NULL
};

readfiles (sub.file_ct.files)
char *sub;
int *file_ct;
struct usersub **files;
{
    struct direct nf;
    struct stat sbuf;
    char fname[BUFSIZE];
    FILE *fd;
    int i, j;
    char *malloc(), *realloc();
    if ((fd = fopen (sub, "r")) == NULL) {
        return (1);
    }
    /* Read the file names into files */
    *file_ct = 0;
while (fread ((char *) &nf, sizeof nf, 1, fd) == 1)
{
    sprintf(fname, "%s/%s", sub, nf.d_name);
    if (nf.d_ino != 0 && stat (fname, &sbuf) == 0
        && (sbuf.st_mode & S_IFMT) == S_IFREG)
    {
        register char **p;
        p = ignore;
        while (**p && strncmp (**p, nf.d_name, DIRSZ))
            ++p;
        if (!*p) {
            if (((*file_ct)++ > 0)
                *files = (struct
                        usersub *)
                realloc
                    ((char *) *files,
                    (unsigned)
                        (sizeof
                            (struct usersub)
                            * (*fil
e_ct))));
            else
                *files = (struct
                        usersub *) malloc
                        ((unsigned)
                            (sizeof
                                (struct usersub)
                                * (*file_ct)))
                ;
        }
        if (*files == NULL) {
            fprintf (stderr,
                "No storage0); exit (1);
        }
        (*files)[(*file_ct)-1]
            .mtime = sbuf.st_mtime;
        strncpy (*files)[(*file_ct)-1].name,
            nf.d_name, DIRSZ);
    }
}
/* Sort the elements of files in decreasing time order */

for (i=1; i<(*file_ct); i++)
    for (j=0; j<i; j++)
        if (files[j].mtime < files[i].mtime)
            struct usersub temp;
            temp = files[i];
            files[i] = files[j];
            files[j] = temp;

/* Clean up */
fclose (fd);
return (0);
#include "art.h"

main()
{
char fname[BUFSIZE], sub[BUFSIZE],
    f[BUFSIZE], comdir[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");

reject(fname,n,sub, f, comdir);
}

*/
reject(fname,n,sub, f, comdir, nm, newfname)
int n;

(}
int getpid();
char letter[BUFSIZE];
char mailinglist[BUFSIZE];
char message[BUFSIZE];
char dirnm[BUFSIZE];
char sys[BUFSIZE];
char date[BUFSIZE];
char auth[10], *getlogin();
struct stat statinfo;
FILE *fopen(), *fp;

sprintf(letter,"ltr\%d", getpid());
sprintf(message,"The file %s in SUBJECT %s was rejected.
    f, sub);

/* Place the information in file letter */
fp = fopen(letter, "w");
puts(message,fp);
close(fp);
40:
41:
42:   if(stat(fname,&statinfo)==FALSE)
43:   {
44:     printf(" %s is not a valid subject",fname);
45:   }
46:   strcpy(auth,(getlogin(statinfo.st_uid)));
47:   fp = fopen(mailinglist, "a");
48:   fputs(auth,fp);
49:   fclose(fp);
50:   mail("letter,mailinglist");
51:   sprintf(sys, "/usr/we/dock/record rejected article
52:   system(sys);
53:   delete(mailinglist);
54:   delete(letter);
55: }
```c
1: 
2: */
3: */ report
4: */ prints article title. s
5: */
6: */
7: report (s,sub,dummy,types)
8: char *s, *types.
9: sub[];
10: { static int first = 1;
11: extern ncount;
12: 
13: if (s) {
14: if (first) {
15: first = 0;
16: switch (*types)
17: {
18: case 'r': printf(" 
review" ,sub);
19: break;
20: case 's': printf(" submission" ,sub);
21: break;
22: default: printf (" 
23: 
24: ncount = 5;
25: 
26: if(ncount%5 == 0) printf(" ");
27: printf ("%15.14s", s);
28: ncount++; 
29: } else if (!first)
30: 
31: 
32: first = 1;
33: 
34: } 
35: }
```
#include "art.h"

/*
 * review(fname, n, 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 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 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, "mig%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 */
mail(letter, mailinglist);
delete(letter);
delete(mailinglist);

/* record for posterity */
sprintf(sys, "usr/we/docx/artrecord
  accepted for review
  system(sys);

/* move the file to new location */
sprintf(message, "mv %s %s", fname, tname);
system(message);

/* need a list of reviewers */
sprintf(message, "%scomments/%s/reviewers", comdir, sub, f);
sprintf(dirnm, "%scomments/%s", comdir, sub);
sprintf(fcheck, "%s/members", dirnm);
mklist(s, dirnm, fcheck);

/* need to make directories for reviewers */
sprintf(message, "mkdirs %s/reviewers
  %scomments/%s/%s, newname, comdir, sub, f");
system(message);

/* first is where to get list, second is dir to build them in */

/* Place the information in file letter */
sprintf(message, "Please add %s as a review subject",
  sub);
fp = fopen(letter, "w");
fpus(message, fp);
fclose(fp);

/* mail directions to all reviewers */

/* mail the info */
mail(letter, s);
}
submit.c

receives as arguments a filename (fname),
types, and an integer n which is the index
into the filename where one wishes to change
?subject to comments.

*/

#include "art.h"

submit(fname, n, sub, f, comdir, nm, newfname)
int n;
{
    char s[BUFSIZE];
    char tname[BUFSIZE];
    char NEWLINE;
    int answer;

    printf( "Do you wish to accept for publication."
    );
    printf( "accept for review, or reject");
    for (answer=0; answer!=TRUE;) 
    {
        printf( "publish, review, or reject:");
        s[0] = ' '; 
        if (gets(s) == NULL) exit(1);
        switch (s[0])
        {
        case 'r': if(strcmp(s,"rev") == 0 ||
        strcmp(s,"revi") == 0 ||
        strcmp(s,"revie") == 0 ||
        strcmp(s,"review") == 0)
        {
            strcpy(tname,fname);
            strcpy(&tname[n],"r",1)
            :
            strcpy(s,"accepted for review");
review(fname, n, sub, f, comdir, tname, nm, newfname);
    answer=TRUE;
break;
}
if(strcmp(s, "reject") == 0 ||
    strcmp(s, "reject") == 0)
    reject(fname, n, sub, f, comdir, nm, newfname);
    answer=TRUE;
break;
}
case 'p':
if(strcmp(s, "p") == 0 ||
    strcmp(s, "pub") == 0 ||
    strcmp(s, "pub") == 0 ||
    strcmp(s, "publish") == 0)
{
    strcpy(tname, fname);
    strcpy(&tname[n], "p", 1);
    strcpy(s, "accepted for publishing");
    publish(fname, n, sub, f, comdir, tname, nm, newfname);
    answer=TRUE;
break;
}
case '?'
    printf(stderr, "-%s%s0,""review",""accepts the article for review by a list of reviewers:"

    printf(stderr, "-%s%s0,""review","")
Tue Jul 17 11:46 1984 submit.c 26-3 /71

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

Tue Jul 17 11:46 1984 typcheck.c 27-1 /1

1: #include "art.h"
2: 
3: 
4: typcheck(artdir, subject, title, types)
char *getlogin(), nm[10], s[BUFSIZE], tmpnm[BUFSIZE], owner[10];

int n, getpw();

struct stat statinfo;

strcpy(nm, getlogin());

n=strlen(nm)-9; /*size of string ssubject*/

strcpy(tmpnm, nm, n);

switch(*types)
{

case 's':
    printf("%ssubjects/%s\n", tmpnm, subject);
    stat(s, &statinfo);
    n=strlen(nm);
    getpw(statinfo.st_uid, &owner[0]);
    return(strcmp(owner, nm));

case 'r':
    printf("%scomments/%s/%s/%s\n", tmpnm, subject, title, nm);
    return(stat(s, &statinfo));

default:
    return(SUCCESS);
}
update

fwrites user_sub structure to file .user_sub
in user's HOME directory

arguments

sub sub[s] to update
sub_ct number of subs
user_sub structure of user subs and
times

/*

#include "art.h"

update (subjects, subject_ct)
struct subjects subjects[MAXBDS];
int subject_ct;
{
struct usersub usersub;
FILE *fopen();
*outfile;
int outcount;
char *getenv();
*homeptr,
home_sub[50];
int i;

if ((homeptr = getenv("HOME")) == NULL )
{
fprintf(stderr,"cannot find HOME
variable0);
exit(1);
}

strncpy (home_sub,homeptr);
strcat (home_sub,"/");
strcat (home_sub,".user_sub");

if((outfile = fopen (home_sub, "w")) == NULL)
{
fprintf(stderr,"art: cannot open
.user_sub0);

	/*
	fwrites user_sub structure to file .user_sub
	in user's HOME directory


times
exit(1);

for(i = 0; i < subject_ct; i++)
{
    strcpy(usersub.name.subjects[i]->name);
    usersub.mtime = subjects[i]->stbuf.st_mtime;
    if(fwrite((char *) &usersub,
              sizeof(struct usersub),1,outfile) != 1)
    {
        fprintf(stderr,"error writing
                    .user_subs0);
        exit(1);
    }
}

fclose(outfile);
/*
outartcls
processes artcl more current than cutoff
arguments
emit article processing function
typically print article or report
title
sub sub processed
cutoff currency time, all artcl more
current than cutoff are "emitted"
assumes we are in artclsubs directory
*/

#include "art.h"

outartcls (emit, sub, art_dir, flag, types)
int (*emit)();
struct subjects *sub;
char *art_dir, *types;
int flag;
{
    struct usersub *artcls;
    int artcl_ct, i;
    long cutoff;
    char full_name[BUF_SIZE];
    /* read article titles and modification times */
    sprintf(full_name, "%s/%s", art_dir, sub->name);
    if (flag == EVERYONE)
    {
        if (readdirs (full_name, &artcl_ct, &artcls)
            == 1) return ;
    }
    else
    {
        if (readfiles (full_name, &artcl_ct, &artcls)
            == 1) return ;
    }
/* process current artcl */
for (i=0; i<artcl_ct; i++) {
    if ((flag == ALL || flag == EVERYONE ||
         sub->new == 1 ||
         artcl->mtime > sub->stbuf.st_mtime) &&
        (typcheck(art.dir.sub->name, artcl->name, types)==SUCCESS))
        (*emit)(artcl->name, sub->name, art.dir, types);
    artcl++;
}
(*emit)((char *) NULL);
fflush (stdout);
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <setjmp.h>
#include <signal.h>
#include <sys/stat.h>
#include <pwd.h>
#include <ctype.h>

#define BAD (substat == subject_ct)
#define NOTHERE (substat == -1)
#define GOOD (substat >= 0 && substat < subject_ct)
#define EVERYONE 2
#define ALL 1
#define NEW 0
#define EXIST 0
#define SUCCESS 0
#define BOARD *(argv)
#define A_BOARD *((argv-1)
#define TITLE *((argv-1)
#define OPTION (*((argv)[0] == ' ' && (*argv)[2] == NULL

/* The number of leading spaces on each line of output */
define INDENT 3
define BUFSIZE 256
define MAXBDS 20
define TRUE 1
define FALSE -1

/*
The following items should not be printed.
*/

extern char *ignore[];

struct usersub
{
    long mtime;
    char name[DIRSIZ];
};

struct subjects
{
struct stat stbuf;
int excl;
int new;
char name[DIRSIZ];
};

extern char stdbuf[BUFSIZ];
extern jmp_buf save_addr;
extern int ncount;
extern long time();
if test ! -f $HOME/.artprofile
then
    cp /usrb/we/dock/src/artprofile.d $HOME/.artprofile
    cat /usrb/we/dock/src/profile.d >>$HOME/.profile
fi
/usrb/we/dock/bin/chgsubs
/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
1: #artsu b.sh shell file
2: #called artsu b
3: 
4: #DIR='pwd'
5: #DEFDIR=/usr/we/clock
6: cd $DEFDIR/comments
7: SUBJECT=""
8: while
9:     test -z "$SUBJECT"
10: do
11:     echo "$SUBJECT":
12:     read SUBJECT
13:     if test -z "$SUBJECT"
14:         then
15:             echo "Enter the name of a subject."
16:             echo "Enter the name of a subject."
17:             echo "of existing subjects.
18:             SUBJECT=""
19:             continue
20:         fi
21: if test "$SUBJECT" == ?
22:     then
23:         echo "Enter the name of a subject."
24:         echo "The following are art names ever"
25:         SUBJECT=""
26:         continue
27:     fi
28: if test -d "$SUBJECT"
29:     then
30:         echo "$SUBJECT already exists as the name of a subject."
31:         echo "Enter the name of a subject."
32:         echo "of existing subjects."
33:         SUBJECT=""
34:         continue
35:     fi
36: done
37: 
mkdir $SUBJECT
38: ../psubjects/$SUBJECT
#!/bin/bash

#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 "$OWNER" = ?
    then
      echo "Enter the login id of the owner."
      echo "The following are valid login ids of valid login ids"
      OWNER=""
      continue
  fi
  if valid "$OWNER"
    then
      echo
      else
      echo "$OWNER is invalid name."
      echo "Enter
      echo "$IDS0
      OWNER=""
      continue
  fi
done
chown "$OWNER" $DEFDIR/psubjects/$SUBJECT
chown $OWNER $DEFDIR/$SUBJECT

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 members of this group."

91: /usr/we/dock/bin/mkmem $DEFDIR/comments/$SUBJECT/members 0

94: cd DIR
ARTDIR=/usr/we/dock

# get user long name
UNAME='uname -n'
MYNAME=`information -name`
if test -z "$MYNAME"
    then MYNAME=${UNAME}:${LOGNAME}
fi

# remember current directory
DIR=`pwd`

cd $ARTDIR/comments

# get sub name and article title
SUBJECT=$1
while test -z "$SUBJECT" -a -z "$SUBJECT"
do
echo "SUBJECT:"
read SUBJECT
if test -z "$SUBJECT"
    then echo "Enter the name of a sub."
        echo "Enter existing subs."
        continue
fi
if test $SUBJECT = ?
    then echo "Enter the name of a sub."
        echo "The following are $ARTDIR/bin/art -e SUBJECT="" $ARTDIR/bin/art -e SUBJECT=""
        continue
fi
if test ! -d "$SUBJECT"

```
tue jul 17 11:46 1984    artsubmit.sh 34-2 /43

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

Tue Jul 17 11:46 1984    artsubmit.sh 34-3 /82
```
if test ! "$TITLE" = "$FILE"
  then   echo Note:
        of article.
  fi

if test -f "$FILE"
  then   OWNER=`ls -1 $FILE | sed "s/ */:/g" | cut
                -f3 `-d:``
        if test ! -w $FILE -o "$OWNER" = "$LOGNAME"
          then   echo
                  you do not have permission to edit it.
                  TITLE=""
        fi
  fi

trp "rm >/dev/null 2>/dev/null /tmp/artfile$$:exit" 1 2 15

cd $DIR

mkdir $ARTDIR/comments/$SUBJECT/$FILE
/usr/we/dock/record "$MYNAME submitted article for
review"

ORIGFILE=""
while test -z "$ORIGFILE"
do
  echo "File to be copied:"
read ORIGFILE
"
if test -z "$ORIGFILE"
    then
        echo "Enter the name of the file to be copied."
        echo "Enter"
        continue
fi

if test "$ORIGFILE" = ?
    then
        echo "Enter the name of the file to be copied."
        echo "it may be a full pathname or a relative path name."
        ORIGFILE=""
        continue
fi

if test ! -f "$ORIGFILE"
    then
        echo The file
        echo "Enter"
        ORIGFILE=""
        continue
fi

done

cp "$ORIGFILE" $ARTDIR/ssubjects/$SUBJECT/$FILE
#arttrack
#arttrack SUBJECT
#arttrack SUBJECT ARETICLE

cd /usrb/we/docX/comments
MYNAME='who am i | awk '{print $1}'
SUBJECT=$1
ARTICLE=$2
ANS=""
VALID=""

while
test -z "$VALID"
do
echo "Do you wish to track articles for which you"
echo "are the author or reviewer, or a subject for"
echo "which you are the owner?"
echo ; echo "type author, reviewer, or owner: "
read ANS

case "$ANS" in
  author|autho|auth|aut|au|a)
    VALID="OK"
    if test -z "$ARTICLE"
      then
        if test -z "$SUBJECT"
          then
            for i in */
              do
                if test -w $1 -a -d $1
                  then
                    echo; echo " SUBJECT/TITLE"
                    echo "$1": echo
                    cat $1/history
                    echo; echo
                  fi
                done
            else
              for i in $SUBJECT/*
                do
                  if test -w $1 -a -d $1
                    then
                      echo; echo " SUBJECT/TITLE"
                      echo "$1": echo
                      cat $1/history
                    done
      done
  esac
esac
then
  echo: echo " SUBJECT/TITLE"
  echo "$i": echo
  cat $i/history
  echo: echo
fi
done
fi
else
  if test -w "$SUBJECT/SARTICLE"
  then
    echo: echo " SUBJECT/TITLE"
    echo "$SUBJECT/SARTICLE": echo
    cat "$SUBJECT/SARTICLE/history"
    echo: echo
  fi
fi

::
owner|own|ow|ow(o)
VALID="OK"
if test -z "$SUBJECT"
then
  for i in *
  do
    if test -w "$i"
    then
      cd "$i"
      for j in *
      do
        if test -d "$j"
        then
          echo; echo " SUBJECT/TITLE"
          echo "$i/$j": echo
          cat "$j/history"
        fi
        done
      cd ..
    fi
done
else
  if test -w "$SUBJECT"
then
cd "$SUBJECT"
for j in *
do
  if test -d $j
  then
echo: echo " SUBJECT/TITLE"
echo "$SUBJECT/$j": echo
cat $j/history
fi
done
fi

::
Reviewer|Reviewer|Reviewer|Review|Review|Review|Re
VALID="OK"
if test -z "$ARTICLE"
then
  if test -z "$SUBJECT"
  then
    for i in */
do
    cd $i
    for j in $MYNAME
do
      echo: echo " SUBJECT/TITLE"
echo "$i": echo
egrep "submitted|accepted|$MYNA ME" history
    echo: echo
    cd ../..
done
done
else
  for i in "$SUBJECT/*
do
  cd $i
  for j in $MYNAME
do
    echo: echo " SUBJECT/TITLE"
echo "$i": echo
egrep "submitted|accepted|$MYNA ME" history
  done
  done
else
Tue Jul 17 11:46 1984  arttrack.sh 35-4 /127

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|$MYNAME" history
139:       echo : echo
140:   cd ../..
141: fi
142: fi
143: esac
144: if test -z "$VALID"
145: then
146:   echo "$ANS is an invalid response"
147: fi
148: done
Tue Jul 17 11:46 1984  chgsubs.sh 37-1  /1

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

cp $HOME/.tmpartprofile $HOME/.artprofile
1: ANSI=""
2: echo here
3: sed -n '1p' try
4: while
5: test -z "$ANS"
6: do
7: echo " keep or remove, keep? "
8: read ANS
9: if test "$ANS" = keep
10: then
11: sed -n d
12: else
13: fi
14: done
Tue Jul 17 11:46 1984  makedirs.sh 41-1 /1

1:
2: for i in 'cat $1'
3: do
4: echo $2/$i
5: mkdir $2/$i'
6: done
Tue Jul 17 11:46 1984

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

1:
2: grep "^$1" $2 | exit 0
3: exit 1
ARTBIN = /usrb/we/dock
CC = cc

ARTOBUJS = art.o process.o given_sub.o
          onintr.o output.o
          mklist.o publish.o
          listcreate.o

get_subs.o outart
initialize.o delete.o edit
reject.o review.o typcheck
readfiles.o

install: art artfile mklist mkmem comment
          sh -x xdock

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

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

artfile.o :
    $(CC) -D artfile.o -o artfile
    chmod 0755 artfile
    cp artfile .. /bin/artfile

mklist : mklist.o mklistcom.o edit.o listcreate.o
    $(CC) -D mklist.o mklistcom.o
    chmod 0755 mklist
    cp mklist .. /bin/mklist

mkmem : art.h mkmem.o listcreate.o edit.o
    $(CC) mkmem.o listcreate.o edit.o
    chmod 0755 mkmem
Tue Jul 17 11:46 1984        Makefile 46-2 /41

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 artsobject
53:    
54:    lpr:
55:      -xpr -l Makefile art.h *.c *.sh | lpr
Appendix 4

Application Code Cross References
<table>
<thead>
<tr>
<th>Function</th>
<th>Line</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>cominfo</td>
<td>3-4</td>
<td>cominfo.c</td>
</tr>
<tr>
<td>delete</td>
<td>5-10</td>
<td>delete.c</td>
</tr>
<tr>
<td>edit</td>
<td>6-9</td>
<td>edit.c</td>
</tr>
<tr>
<td>getsubs</td>
<td>7-17</td>
<td>getsubs.c</td>
</tr>
<tr>
<td>given_sub</td>
<td>8-14</td>
<td>given_sub.c</td>
</tr>
<tr>
<td>initialize</td>
<td>9-17</td>
<td>initialize.c</td>
</tr>
<tr>
<td>listcreate</td>
<td>10-17</td>
<td>listcreate.c</td>
</tr>
<tr>
<td>mail</td>
<td>11-2</td>
<td>mail.c</td>
</tr>
<tr>
<td>main</td>
<td>1-31</td>
<td>main.c</td>
</tr>
<tr>
<td>main</td>
<td>2-9</td>
<td>mainfile.c</td>
</tr>
<tr>
<td>main</td>
<td>4-16</td>
<td>comment.c</td>
</tr>
<tr>
<td>main</td>
<td>13-15</td>
<td>mklistcom.c</td>
</tr>
<tr>
<td>main</td>
<td>14-16</td>
<td>mkmem.c</td>
</tr>
<tr>
<td>mklist</td>
<td>12-16</td>
<td>mklist.c</td>
</tr>
<tr>
<td>newcount</td>
<td>15-18</td>
<td>newcount.c</td>
</tr>
<tr>
<td>onintr</td>
<td>16-18</td>
<td>onintr.c</td>
</tr>
<tr>
<td>outartcls</td>
<td>18-19</td>
<td>output.c</td>
</tr>
<tr>
<td>output</td>
<td>17-1</td>
<td>output.c</td>
</tr>
<tr>
<td>print_item</td>
<td>18-6</td>
<td>print_item.c</td>
</tr>
<tr>
<td>process</td>
<td>19-4</td>
<td>process.c</td>
</tr>
<tr>
<td>publish</td>
<td>20-15</td>
<td>publish.c</td>
</tr>
<tr>
<td>readdir</td>
<td>21-26</td>
<td>readdir.c</td>
</tr>
<tr>
<td>readfiles</td>
<td>22-26</td>
<td>readfiles.c</td>
</tr>
<tr>
<td>reject</td>
<td>23-17</td>
<td>reject.c</td>
</tr>
<tr>
<td>report</td>
<td>24-7</td>
<td>report.c</td>
</tr>
<tr>
<td>review</td>
<td>25-4</td>
<td>review.c</td>
</tr>
<tr>
<td>submit</td>
<td>26-13</td>
<td>submit.c</td>
</tr>
<tr>
<td>typcheck</td>
<td>27-4</td>
<td>typcheck.c</td>
</tr>
<tr>
<td>update</td>
<td>28-15</td>
<td>update.c</td>
</tr>
</tbody>
</table>

```c
__exit__

initialize 9-10 extern __exit__();

9-13 signal (SIGQUIT, __exit__);

chdir. cominfo 3-12 chdir(s);

cominfo #DEFN 3-4 "<DEFN>" cominfo(dirnm,nm)

print_item 18-48 cominfo(newfname,nm);

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

ctime print_item 18-32 char *ctime();
```
review 25-43 delete(letter):
delete(mailinglist):

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

exit get_sub 7-37 exit(1):
      43 exit(1):
main  1-103 if (runcmd == NO) exit(1):
      119 exit(1):
      32 if (++k == 14) exit(0):
      25 if (gets(s) == NULL) exit(1):
      62 exit(0):
      22 case 'q': exit(1):
print_list 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):
      42 exit(1):
      51 exit(1):

fclose edit 6-41 fclose(fp):
      42 fclose(tmp):
listcreat 10-54 fclose(fp):
main  4-35 fclose(fp):
print_list 18-52 fclose(fd):
publish 20-35 fclose(fp):
      46 fclose(fd):
readdirs 21-86 fclose(fd):
readfiles 22-85 fclose(fd):
reject 23-38 fclose(fp):
      50 fclose(fd):
review 25-28 fclose(fp):
      39 fclose(fp):
fclose(fp);
fflush(stdout);
while(fgets(s, sizeof s, fp) != NULL)
stat(fd, &buf);
fopen(fp);
if((fp = fopen(fname, "r")) == NULL)
if((fp = fopen(fname, "a")) == NULL)
print(len(fd = fopen(fname, "w")) == NULL)
FILE *fp = fopen(letter, "w");
FILE *fp = fopen(mailinglist, "w");
FILE *fp = fopen(mailinglist, "a");
FILE *fp = fopen(letter, "w");
FILE *fp = fopen(letter, "w");
FILE *fp = fopen(letter, "w");
FILE *fp = fopen(home_subs, "w") == NULL);fprintf(stderr,"art: name for ARTDIR0, art_dir):fprintf(stderr,"art: cannot read
fprintf(get_subs)
print(len(printer = fopen(review, next, quit)
-%10s%0,"review","print article
-%10s%0,"review","print article
"next","go to next article");
18- 72 fprintf(stderr," %-10s%0.
"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",subjects[i]
->name);
19- 170 fprintf(stdout,"existing subjects
:*"):
19- 175 fprintf(stdout,"%-15.14s",subjects[i]
->name);
19- 180 fprintf(stdout," 0.sub):
19- 215 fprintf(stderr,"art:
"option0.option");
readirs 21- 65 fprintf(stderr, "No storage0);
readfiles 22- 64 fprintf(stderr, "No storage0);
submit 26- 70 case 't':
    fprintf(stderr, "-10's%0. review, accepts the
    fprintf(stderr, "-10's%0.
    "publish", accept for publication.
    fprintf(stderr, "-10's%0.
    "reject", reject the article, the author
    fprintf(stderr, "unknown response, please try again.01:\n
update 28- 30 fprintf(stderr,"cannot find HOME
    variable0);
    fprintf(stderr,"art: cannot open
    .user_subso);
    fprintf(stderr,"art: error writing
    .user_subso);

fputs edit 6- 37 case 'k':
    fputs(s.tmp);
    fputs('0.tmp');
listcreat 10- 46 fputs(s.fp);
    fputs('0.fp');
main 4- 31 fputs(s.fp);
    fputs('0.fp');
publish 20- 34 fputs(message.fp);
    fputs(auth.fp);
reject 23- 37 fputs(message.fp);
    fputs(auth.fp);
review 25- 27 fputs(message.fp);
    fputs(auth.fp);
    fputs(message.fp);
    fputs(message.fp);
fnread readirs 21- 44 while (fnread ((char *) &nf, sizeof
    nf, 1, fd) == 1) {
    readfiles 22- 43 while (fnread ((char *) &nf, sizeof
    nf, 1, fd) == 1) {
fsstat printrite 18- 37 fsstat (fileno (fd), &buf);
fwrite update 28- 48 if( fwrite ((char *) &usersub,
    sizeof(struct usersub), 1, outfie) != 1)
get_sub #DEFN 7- 17 DECLARE >> get_sub (art_dir, art_subs, art_excl, subjects, subjct_ct)

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

main 1- 109 get_sub (art_dir, art_subs, art_excl, subjects, subjct_ct);
getenv initialize 9- 11 char *env, *getenv();
    env = getenv("ARTSUBS");
update 23 char *getenv();
26-  26 if ((homeptr = getenv("HOME")) == NULL )

getuid main 13- 16 whoami=getuid();

getlogin dominfo 3-  7 char *getlogin(), s[BUFSIZE],
    sys[BUFSIZE];
20- 10 strcpy(sys, getlogin());
publish 20- 24 char auth[10], *getlogin();
20- 43 strcpy(auth,(getlogin(statinfo.st_uid)));
reject 23- 26 char auth[10], *getlogin();
23- 47 strcpy(auth,(getlogin(statinfo.st_uid)));
review 25- 17 char auth[10], *getlogin();
25- 36 strcpy(auth,(getlogin(statinfo.st_uid)));
typcheck 27-  7 char *getlogin(), rm[10], s[BUFSIZE]
    , tmpnm[BUFSIZE], owner[10];
27- 10 strcpy(tmpnm,getlogin());

getpid edit 6- 16 int getpid();
6- 19 sprintf(tmofile, "tmp%d", getpid());
publish 20- 19 int getpid();
20- 28 sprintf(letter, "1tr%d", getpid());
20- 36 sprintf(mailnglist,"mig%d",getpid());
reject 23- 21 int getpid();
23- 32 sprintf(letter, "1tr%d", getpid());
23- 46 sprintf(mailnglist,"mig%d",getpid());
review 25-  8 int getpid();
25- 21 sprintf(letter, "1tr%d", getpid());
25- 31 sprintf(mailnglist,"mig%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);
      6-  29 gets(ans);
      6-  35 gets(ans);

listcreat  edit  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  print_itc  12-  24 gets(s);
     12-  28 gets(s);
     12-  56 gets(s);

submit  18-  58 gets(s);
     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(adopt, 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 /initialize

initialize #DEFN  9-  7 <<DEFN>> initialize (art_dir,
main  1- 108 initialize (art_dir, art_subs, art_opts, art_excl, art_excl.ptypes.opti)
isalnum main 2- 24 if(isalnum(argv[1][j])) |

isupper process 18- 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 subjects[*subject_ct] = malloc(
        sizeof (struct subjects));
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")
    25- 59 mklist(s, dirn, fcheck);
newcount #DEFN 15- 5 <<DEFN>> newcount(s, sub)
process 19- 11 int print_item(), report(), count()
    18- newcount();
19- 104 outartcils(newcount.subscts[stat].art_dir.NEW.types);
19- 111 outartcils(newcount.subscts[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)

outartcils #DEFN 29- 18 <(DEFN) outartcils (emit.sub.
main 1- 115 outartcils(print_item,subscts[i].art_dir,flag.types)
process 19- 30 outartcils(report,subscts[stat].art_dir.NEW.types);
19- 38 outartcils(report,subscts[i].art_dir.NEW.types);
19- .55 outartcils(report,subscts[stat].art_dir.ALL.types);
19- 63 outartcils(report,subscts[i].art_dir.ALL.types);
19- 60 outartcils(report,subscts[stat].art_dir.EVERYONE.types);
19- 87 outartcils(report,subscts[i].art_dir.EVERYONE.types);
19- 104 outartcils(newcount,subscts[stat].art_dir.NEW.types);
19- 111 outartcils(newcount,subscts[i].art_dir.NEW.types);
19- 129 outartcils(print_item,subscts[stat].art_dir.ALL.types);
19- 139 outartcils(print_item,subscts[i].art_dir.ALL.types);
19- 191 outartcils(print_item,subscts[stat].art_dir.NEW.types);

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

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

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

main 1- 28 int print_item();
1- 115 outartcils(print_item,subscts[i].art_dir.NEW.types);
process 19- 11 int print_item(). report(). count()

Tue Jul 17 11:46 1984 X-REF -- FUNCTIONS 9 /newcount
19- 129 outartcls(print_item.subjects[subst t].art_dir.ALL.types);
19- 139 outartcls(print_item.subjects[1].
   art_dir.ALL.Types);
19- 191 outartcls(print_item.subjects[subst t].art_dir.NEW.types);
19- 199 print_item ( item. sub.art_dir);
19- 206 print_item ( item. subjects[1]->name,
   art_dir);

3- 15 printf(" cominfo
   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
   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
   Type a carriage
   return when ready for the file.");
3- 23 printf(" unable to remove list.");

5- 14 printf(" unable to remove list.");

6- 21 printf(" Unable to open %s",
   tmpfile);
6- 23 printf(" Unable to open %s",fname);
6- 28 printf(" 89s: keep or delete
   [keep]?", s);
6- 32 case 'i': printf(" keep:
   entry remains in list");
6- 33 printf(" delete: removes entry
   from list");
6-  34 printf("96s: keep or delete
[keep]? ", s);
listcreate 10-  18 printf("can't open file\n");
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(ILLOG, argv[1], USE);
1-  87 printf(ILLOG, ptypes, tempt, USE);
1-  95 printf(ILLOG, 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 carriage
return.");
4-  28 printf(" ");
13-  19 printf("Lists are created on a
subject basis by the owner of the group.");
13-  22 printf("subject: ");
13-  29 printf("Enter 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.");

Tue Jul 17 11:46 1984  X-REF -- FUNCTIONS  12 /printf

13-  38 printf("%s is not a valid
subject\n");
13-  46 printf("You are not the owner of
the group %s\n");
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("Edit: examine
       the entries one at a time");
14-  24 printf("Quit: leave the list as
       it is.");
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 subject? ")
12-  27 case 'y':   printf(" list: ");
12-  55 printf(" Do you wish to edit the
       file or quit?");
18-  47 printf(" subject: %s
       article: %s, sub, f");
19-  163 if((j++)%5 == 0) printf(" ");
19-  174 if((j++)%5 == 0) printf(" ");
19-  194 if(ncount == 0) printf("There are
       no new article on
19-  200 if(ncount == 0) printf(" not a article on
19-  210 if(ncount == 0) printf("There is no
       article called
20-  41 printf(" %s is not a valid
       subject, fname");
reject 23- 44 printf(" %s is not a valid
subject",fname):
report 24- 19 case 'r':printf(" 
review":"sub):
24- 21 case 's':printf(" 
submission":"sub):
24- 23 default: printf(" 
24- 27 if(count%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(subjects,subject_ct,
art_dir,sub_item,option,types)
main 1- 130 process(subjects,subject_ct,art_dir,
"",",option,types):
1- 136 process(subjects,subject_ct,art_dir,
adopt,"",option,types):
1- 151 process(subjects,subject_ct,art_dir,
sub",",",types):
1- 154 process(subjects,subject_ct,art_dir,
sub,adopt,"",types):
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[1][j],stdout);
2- 38 putc(' ',stdout);
2- 41 putc('0',stdout);
print_lte 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_lte 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)

readdir #DEFN 21-26 <DEFN> readdir (sub_file_ct, 
files)
outartcls 29-33 if ( readdir (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 22-36 char *malloc(), *realloc();
22-54 realloc ((char *) *files, 
reject #DEFN 23-17 <DEFN> reject (fname, n_sub, f, 
comdir, nn, newfname)
submit 26-48 reject (fname, n_sub, f, comdir, nn, 
newfname):

report #DEFN 24-7 <DEFN> report (s, sub, dummy, types)
process 19-11 int print_item(), report(), count() 
newcount();
19-30 outartcls (report, subjects[stat], 
art_dir.NEW.types);
19-38 outartcls (report, subjects[i], art_dir, 
NEW.types);
19-55 outartcls (report, subjects[stat], 
art_dir.ALL.types);
19-63 outartcls (report, subjects[i], art_dir, 
ALL.types);
19-80 outartcls (report, subjects[stat], 
art_dir.EVERYONE.types);
19-87 outartcls (report, subjects[i], art_dir, 
EVERYONE.types);
Tue Jul 17 11:46 1984  X-REF -- FUNCTIONS 15 /review

review  xDEFN 25-  .4 "<DEFN> review(fname, n, sub, f, comdir, tname, nm, newname)
submit 26- 39 review(fname, n, sub, f, comdir, tname, nm, newname):

setjmp print_ite 18- 42 if (setjmp(save_addr))
signal initialize 9- 12 if (signal(SIGQUIT, SIG_IGN) ==
19- 13 signal(SIGQUIT, _exit);
print_ite 18- 44 if (signal(SIGINT, SIG_IGN) ==
18- 45 signal(SIGINT, onintr);
18- 53 if (signal(SIGINT, 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
16- 18 sprintf(tmpfile, "tmp%0d", getpid());
6- 43 sprintf(s, "mv %s %s", tmpfile, fname)
get_sub5 7- 54 sprintf(fullname, "%s/%s", art_dir, dirbuf, d_name);
listcreat 10- 43 sprintf(sys, "~/usr/we/dock/v
3- 11 6 sprintf(com, "~/usr/we/dock/ml
11- 71 sprintf(subnames + strl(subnames),
13- 35 sprintf(dirmame, "%s/comments/%s", ARTDIR.s);
13- 59 sprintf(sys, "rm -f %s/%s", dirname.s);
13- 66 sprintf(fname, "%s/%s", dirname.s);
12- 29 sprintf(e, "%s/%s", dirname.s);
12- 33 sprintf(s, "cp %s %s", e, fname);
outartcls 29- 30 sprintf(fullname, "%s/%s", art_dir, sub->name);
output 17- 6 sprintf(t, "~/usr/ccc/bin/vi %s %s", filename, outs);

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

print_ite 18- 25 sprintf(fname, "%s/%s/%s", art_dir, sub, f);
18- 26 sprintf(newfname, "%s/comments/%s/%s", comdir, sub, f);
publish 20- 28 sprintf(letter, "1tr%d", getpid());
20- 29 printf(message,"The file %s in
   SUBJECT %s was accepted
   ");
20- 55 printf(sys,"/usr/we/dock/re
cord
20- 59 printf(message,"mv %s %s", fname,
tname):
readdirs
21- 45 printf(fname,"%s/%s",sub,nf.d_name)
: readfiles
22- 44 printf(fname,"%s/%s",sub,nf.d_name)
: reject
23- 32 printf(letter,"ltr%d",getpid());
23- 33 printf(message,"The file %s in
   SUBJECT %s was rejected.
   ");
23- 46 printf(mailinglist,"mil%d",getpid());
23- 53 printf(sys,"/usr/we/dock/re
cord
review
25- 21 printf(letter,"ltr%d",getpid());
25- 22 printf(message,"The file %s in
   SUBJECT %s was accepted
   ");
25- 48 printf(sys,"/usr/we/dock/ar
trecord
25- 52 printf(message,"mv %s %s", fname,
tname):
25- 56 printf(s,"%scomments/%s/%s/reviewes",cmdir,sub,f);
25- 57 printf(dirrm,"%scomments/%s",
   cmdir,sub):
25- 58 printf(fcheck,"%s/members", dirrm):
25- 62 printf(message, "mkdirs %s/reviewe
   rs %scomments/%s/%s", newfname, comdir, sub, f):
25- 69 printf(message,"Please add %s as a
   review subject",sub);
typcheck 27- 15 case 's':    sprintf(s,
    "%ssubjects/%s.tmppn.subjectt);  
27- 20 case 'r':    sprintf(s,
    "%scomments/%s/%s.tmppn
stat  get_subs 7-  33 if(stat(dir, &stbuf) == -1) ||
7-  55 if(stat(fullname, &stbuf) == -1)  
    continue;
7-  59 stat(fullname, & (subjs[subject-
    ct]->stbuf));
main 13-  36 if(stat(dirname, &statinfo)==FALSE)
mklist 12-  17 if((stat(filename, &statinfo)==FALSE)
    && (dirname == '.'))  
12-  20 if(stat(a, &statinfo)==FALSE)
12-  46 if(stat(filename, &statinfo)==FALSE)
publish 20-  39 if(stat(filename, &statinfo)==FALSE)
readdirs 21-  46 if (nf.d_ino ! = 0 && stat (fname,
    &stbuf) > 0)
readfiles 22-  45 if (nf.d_ino != 0 && stat (fname,
    &stbuf) > 0)
reject 23-  42 if(stat(filename, &statinfo)==FALSE)
review 25-  32 if(stat(filename, &statinfo)==FALSE)
typcheck 27-  16 stat(s, &statinfo);  
27-  21 return(stat(s, &statinfo));
strcat initialize 9-  17 strcat(dir, "/");
9-  20 strcat(dir, "comments");
9-  23 strcat(dir, "types");
9-  24 strcat(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[1][1]))
    == 0)
1-  77 if (((tempt=strchr(typ.argv[1][1]))
    == 0)
strncpy get_subs 7-  51 if(strncmp(dirbuf.d_name, ".", 0) == 0)
7-  52 if(strncmp(dirbuf.d_name, "...", 0) == 0)  
    continue;
7-  82 if(strncmp(subjects[i]->name.env) ==
Tue Jul 17 11:46 1984 X-REF -- FUNCTIONS 18 /strncpy
7-  99 if(strncmp(subjects[i]->name.env) == 0)
given_sub 8-  27 if(strncmp(subjects[i]->name.sub) == 0) return(i);
initializ 9-  19 if(strncmp(types, "p")==SUCCESS) &&
```c
#include <stdio.h>
#include <string.h>

int main()
{
    char *option = "e";
    int opt = 0;
    char *rev = "rev";
    char *pu = "pu";
    char *pub = "pub";
    char *pub1 = "pub1";
    char *publ = "publ";
    char *publish = "publish";
    char *opt = "opt";
    char *AGOCEN = "AGOCEN";
    char *rsRS = "rsRS";
    int GOOD = 1;

    char *art_dir = ARTDIR;
    char *art_subs_env = ART_SUBS_ENV;
    char *art_opts_env = ART_OPTS_ENV;
    char *art_excl_env = ART_EXCL_ENV;

    char *statinfo = statinfo;
    char *statinfo_st_uid = statinfo_st_uid;
    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

    char *argv[3] = {"arg1", "arg2", "arg3"};
    char **argv_end = &argv[3];

    char *argv0 = argv;
    char *argv1 = argv + 1;
    char *argv2 = argv + 2;
    char *argv3 = argv + 3;

    int argc = 3;
    int argvn = 3;

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

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

strlen initalize 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[1]);
print_ite 18- 23 n=strlen(art_dir) - 9; /*size of
string subjects*/
readdir 21- 50 while (*p & strncmp (*p, nf.d_name, len(*p)))
typcheck 27- 11 n=strlen(artdir) - 9; /*size of
string subjects*/
27- 17 n=strlen(nm);

strncmp main 1- 68 if(strncmp(argv[1], "+", 1)!
return;
print_ite 18- 66 case 'q':
if (strncmp(ptypes,
"s", 1)!
submit(fname, n, sub.f,
process 19- 156 if (!(NOTHERE) & (strncmp(ptypes,
"p", 1))!=SUCCESS))
19- 168 if (!(NOTHERE) & (strncmp(ptypes,
"p", 1))!=SUCCESS))
readdir 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);
readdir 21- 69 strncpy (*files)[(*file_ct)-1]
.name,
strtok  get_sub  7-  28 char *env, *strtok();
7-  76 env = strtok(arg_sub, "");
7-  88 env = strtokO,"");
7-  93 env = strtok(arg_sub, ":" );
7- 105 env = strtokO, ":" );
main  1- 133 sockfd = strtok(subnames, ":");
1- 137 sockfd = strtokO,"";)
1- 145 sockfd = strtok(subnames, ":" );
1- 149 sockfd = strtokO,"";)
1- 155 sockfd = strtokO,"";)

submit  #DEFN  26-  13 <<DEFN>> submit(filename, subf, comdir, rm, newfilename)
print_ite  18-  66 case 'q':  if strncmp(ptypes,
"s",1)== 0) submit(filename, subf);

system  cominfo  2-  14 system(sys);
edit  6-  44 system(s);
listcreat  10-  44 if (system(sys) == SUCCESS)
mail  11-  7 system(com);
main  13-  60 system(sys);
mklist  12-  34 system(s);
output  17-  7 system(t);
publish  20-  56 system(sys);
reject  20-  60 system(message);
review  23-  54 system(sys);
time  25-  49 system(sys);
main  25-  53 system(message);
25-  63 system(message);

time  main  1-  40 long time();
1- 116 subject[i]->stbuf.st_mtime = time
((long *) 0);
process  19- 10 long time();
19- 130 subject[substat]->stbuf.st_mtime = 
time ((long *) 0);
19- 140 subject[i]->stbuf.st_mtime = time
((long *) 0);
Tue Jul 17 11:46 1984 X-REF -- FUNCTIONS 21/time

19-192 subjects[subject]->stbuf.st_atime =
time((long*)0);

typcheck #DEFN 27-4 <<DEFN>> typcheck(art_dir,subject,
title,types);

tyartc1s 29-45 (tycheck(art_dir,sub->name,
artc1s->name,types)==SUCCESS)

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

update #DEFN 28-15 <<DEFN>> update (subjects,subject_ct)

main 1-118 update (subjects,subject_ct);

process 19-131 update (subjects,subject_ct);
19-142 update (subjects,subject_ct);
19-193 update (subjects,subject_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,
subjects,subject_ct)
7-18 char art_dir[],art_subs[],art_excl[]
7-  33 if(statdir(&stbuf) == -1 ||
7-  36 fprintf(stderr,"art:
    name for ARTSUBSOARTDIR):;
7-  40 if((fd = open(art_dir, O)) == -1)
7-  42 fprintf(stderr,"art: cannot read
7-  54 fprintf(fullname, "%s/%s", art_dir,
        dirbuf.q_name);
8-    initialize(art_dir,art_subs,
        art_opts,art_excl,types,option)
8-     char art_dir[],art_subs[],art_opts[
        .art_excl[],"types,option;
8-     strcpy(art_dir,ARTDIR);
8-     strcat(art_dir,"/");
8-     strcat(art_dir,"comments");
8-     strcat(art_dir,"subjects");
9-   108 initialize (art_dir,art_subs,
    art_opts,art_excl,types,option):
9-   108 get_subs (art_dir,art_subs,art_excl,
     subjects,&subject_ct):
10-  115 outartcls(print_item,subjects[i],
     art_dir,NEW,types);
10-  130 process(subjects,subject_ct,art_dir,
     "","option,types);
10-  136 process(subjects,subject_ct,art_dir,
     soopt,"",.option,types);
10-  151 process(subjects,subject_ct,art_dir,
     sub","",.types);
10-  154 process(subjects,subject_ct,art_dir,
     sub,soopt,"",.types);
12-  29 outartcls (emit,sub,art_dir,flag,
     types)
12-   2! char *art_dir,*types;
12-   30 fprintf(fullname,"%s/%s",art_dir,
     sub->name);
12-   45 (typcheck(art_dir,sub->name,
     artcls->name.types)==SUCCESS) )
art_excl #GLOBAL
get_subs 7
get_subs (art_dir, art_sub, art_excl, subjects, subject_ct)
initialize 7
initialize (art_dir, art_sub, art_opt, art_excl, types, option)

main 1
initialize (art_dir, art_sub,
art_opts #GLOBAL
1-  17 char art_opts[BUFSIZE];
initialize 9-  7 initialize(&art_dir, &art_sub[],
art_opts, &art_excl.types.option)
9-  8 char art_dir[], art_sub[], 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_sub[],
art_opts, &art_excl.types.option);

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

#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- 136 ncount = 0;
19- 194 if(ncount == 0) printf("There are
19- 200 if(ncount == 0) printf(
    "no new artcl on
19- 210 if(ncount == 0) printf("There is no
    "article called
report 24- 12 extern ncount;
24- 25 ncount = 5;
24- 27 if(ncount%5 == 0) printf(" ");
24- 29 ncount++;

#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))
"%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, " ");
ALL art.h 30- 14#define ALL 1
ARTDIR initialize.c 9- 1#define ARTDIR "*/usrb/ws/dock
A_BOARD art.h 30- 19#define A_BOARD *(argv
BAD art.h 30- 10#define BAD (subst
ARTDIR mklistcom.c 13- 2#define ARTDIR "*/usrb/ws/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
subjects art.h 30- 40 struct subjects
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' comments are controlled.