CHI 24-MAY-74 16:09 23118

Message to JGR about our current NLS plans

Jeff Rothenburg sent me a rush message asking for answers to several questions. I wrote this back to him. Please feel free to SNDMSG any corrections or omissions.

CHI 24-MAY-74 16:09 23118

Message to JGR about our current NLS plans

Jeff, very sorry I did not get this to you sooner. Hope this is still of some use to you. 1

1. What is the current status and proposed direction of your on-line help facility? What short- or long-term effect do you see for Laura Gould's work, in terms of becoming incorporated into NLS as part of its on-line capabilities ? 2

Current status of NLS help facilities: 2a

Now in use in the running NLS at ARC. Will be released to Office-1 users sometime this summer. 2a1

If the syntactic/semantic help provided by the command language interpreter (based on the grammar) is insufficient, the user may type "Q. This loads the help data base and presents the user with a brief description of the users current context and a menu of possible things he might want to know about that particular command context. In addition to selecting any of the menu items for further elaboration, the user may also request explanations of terms such as VIEWSPECS, STRUCTURE, SYNTAX CONVENTIONS, etc. He may also request help with the help system. 2a2

The primary problem with the current system is that there is an unreasonably long pause (perhaps 30 seconds) when the user types "Q, while the help data base is loaded and searched for a reasonable match to his current command state. We hope to improve this before releasing it to Office-1 users. 2a3

We tried bringing the system up at ARC without training our users and encouraged them to learn about the new system by using the various help features. These user's were all familiar with the old system. This was a minor success, in that most users stumbled their way along, but many felt that it was a very tedious way to learn a system; that a formal training session together with written documentation combined with the online help features would have been perfect, but the restriction to only online facilities tried their patience. We have not tried the same thing with inexperienced users. 2a4

Laura Gould's work: 2b

If the work that Laura is doing proves to be sufficiently responsive to the user's needs and seems to be an effective training aid, I think there is no doubt that we would try to incorporate it. I think that only time will tell, but I have high hopes that she will come up with something reasonable. (I have similar hopes with regard to your work.) 2b1

Message to JGR about our current NLS plans

2, To what extent do you see NLS and the Journal as a replacement or supplement to SNDMSG? 3

We view the continued development of the SNDMSG/READMAIL facilties as well as your work to be attempts to provide almost exactly the same facilities that NLS and the Journal attempt to provide (although we view NLS as having a broader range of applications than just message sending!). We fully expect that when BBN and you get further along in your respective developments, you will start receiving exactly the same criticisms that we receive about our system being too complicated, too hard to learn, etc. -- you will begin to perceive why all those facilities are really desireable but no one else will. Please do not misunderstand me here. We feel that there is plenty of room for independent development here, but mearly wish that there were more explicite recognition (and thus cooperation) that we are all working on the same problem, though perhaps with slightly different emphases and motivations. 3a

The new Journal system now also supports unrecorded mail, and thus can be even more like SNDMSG. It is also as easy to use and provides more extensive features. It informs users who are logged in at the time of mail delivery that they just got some mail. Mail is delivered into an ACTION mailbox or into an INFOMATION ONLY mailbox (specified by sender). The Journal also supports private messages (only sender and recipients can read). 3b

We also have a READMAIL subsytem partially implemented. This allows the user to sort his mail into different catagories, forward mail on to other users or groups of users, request that an item be re-sent to you at a certain time (a reminder feature), etc. 3c

3, What are your current and intended plans for disseminating NLS to the "real world" ? What sites are currently using it ? In what form are they using it (TNLS, DNLS, restricted subset, etc.) ? 4

The Office-1 facility was set up explicitly as a vehicle for transferring NLS out to "real" users in a way that we could both control and observe. This has been very successful so far. We have a staff explicitly assigned to handle the needs of those users and to learn how to transfer this sort of system more effectively. The office-1 system supports both DNLS and TNLS. In addtion, there is a version of NLS (minus the Journal) at PARC and at BBN. The one at PARC is used quite a lot (TNLS only, I beleive); the one at BBN (TNLS) little, to my knowledge (although several people there are interested in using it in their day-to-day work). 4a

Message to JGR about our current NLS plans

There has been a running disagreement here about whether or not NLS should be released as part of standard TENEX. I happen to favor this move, but feel that it requires improved hardcopy documentation. Others here are very paranoid about it (afraid people will have trouble learning it and will thus give it a bad name) or fear that it will jepardize the success of office-1, office-2, etc. 4b

4. Do you see any problems with the structured-text approach with respect to disseminating NLS to non-computer people ? 5

We are currently defining a user interface that does not utilize the structure facilities. We will have to wait and see how well it works. We have had little trouble with structure so far, since most users have been exposed to outlines in high school and college writting courses. 5a

Perhaps I should point out that structured text is used in NLS because of a belief that it can help people externalize their ideas more clearly. The use of such structure has a profound effect on one's work, I can assure you of that. 5b

5. To what extent is the NLS interface now tailored to individual users ? Is this done at all dynamically ? Is there any automatic adaptation of the interface to the user's style, or is it rather a matter of the user setting things up explicitely to suit himself? 6

The NLS user interface can now be tailored to individual users in the following ways: 6a

command word recognition 6a1

The scheme used by the system to allow a user to choose between alternative command words in a command. For example, what must the user do to choose one of the following: "COPY", "CONNECT", "DELETE", and "DISCONNECT" ? 6a1a

We now support four different disciplines 6a1b

1) DEMAND (like TENEX exec uses), 6a1b1

2) ANTICIPATORY (minumum unique string of characters -- DE for DELETE, DI for DISCONNECT), 6a1b2

3) FIXED (three characters required -- COP for COPY, CON for CONNECT, DEL for "DELETE", DIS for "DISCONNECT"), and 6a1b3

4) EXPERT or FIRST CHARACTER (frequently used commands

are recognized by first letter, other commands by one of the above schemes after typing a SPACE character). 6a1b4

amount and type of prompting 6a2

prompt user for next type of input? prompt user for optional information? 6a2a

amount and type of command feedback 6a3

echo noise words to the user ? echo rest of command word to user after command word has been recognized ? Apply an upper bound to length of command word/noise word echoed to user ? 6a3a

keyboard character translations 6a4

for each type of terminal supported, user may request that certain keys be translated into NLS control characters, such as Command Accept, Command Delete, etc. 6a4a

use of structured-text 6a5

Does the user want to deal with structured-text ? 6a5a

use of viewspecs filters 6a6

Does user wish to deal with viewspecs ? 6a6a

default viewspecs and tabs 6a7

allows user to specify the view parameters to use when starting NLS 6a7a

list of commands automatically executed at startup time 6a8

Allows user to specify a set of NLS commands to be automatically invoked when user enters NLS. 6a8a

location of statement name directory 6a9

Allows user to specify a list of external NLS statement names to be searched if the user requests to see a statement by some name, whereever it is, and no statement by that name exists in the file specified by the user. The list of external statement names consists of a set of ordered pairs of the form (NAME, LINK). The link is a pointer to where this name is to be found in some file. 6a9a

Message to JGR about our current NLS plans

Planned but not yet implemented 6b

proficienty level (probably on subsys basis) 6b1

This will allow a user (or his trainer) to specify that, say for each subsystem, he is at proficiency level N. This will allow him to use and see some commands and system features, but not others. Note that this impact the help features. 6b1a

subsystems available 6b2

This will allow a user to specify which subsystems he wants available to him and which he is given when entering NLS. 6b2a

menu selection (may not happen) 6b3

this is a new scheme for allowing the user to choose between alternatives by bugging the one he wants. 6b3a

At present, none of the user-profile attributes are changed dynamically, although this would be trivial to do -- the hard part is determining when to change it and to what (without shaking the user up too much). We have no immediate plans to modify the user-profile dynamically. We hope that some good things will be learned about that by you and Laura Gould. Of course, if you wished to use NLS as a vehicle for doing so, we would be most happy to cooperate in any way we can. In the present system, the user may change his profile at will, or a trainer may change it for him. 6c

6. To what extent are you interested in providing alternate language forms (e.g. your "expert mode", etc.) ? 7

We are VERY interested in providing and investigating the effectiveness of alternate command language interfaces too the user. The user-profile things, proficiency levels and the ability to define new command languages easily are all examples of our investment in the ability to do just that. 7a

7. Does NLS have any "level" structure in the sense of providing a simplified subset of itself for beginging users ? Do you see that as a desireable way to go in the future ? 8

Our user training program is now organized into such levels. By the time the system is released to office-1, we hope to have added this facility to NLS itself (see proficiency level above). There is considerable debate here as to the desireability of this approach. We intend to try it and find out. In addition, we may try providing user interaces that take advantage of local jargin

for some groups of users to ease their transition into using this type of tool. (You should have figured out by now that we are interested in providing people with tools, functional capabilities that we think can make a significant difference in terms of their effectiveness. We are pushing no particular command language down their throats. We will bend it however we need to to get them to use it , so long as we still believe that the resulting usage will make a significant difference in their work.) 8a

8. What would you say are your major goals for the future expansion, improvement, extension, change ? 9

coupling NLS to a large variety of other tools, such as data management systems, statistical analysis packages, project management packages, 9a

increased system responsiveness through use of satellite processors (frontends), movement of NLS backend (file manipulation) into other operating systems and other machines (IBM, CDC, Burroughs, etc.) and running totally on mini's, 9b

File system changes to allow storing many different types of data in the file (text, numerical, grahics, format controls, voice data) 9c

reinstating line drawing graphics, formal constructs in CML for defining windows and their usage, 9d

effective use of portable and microfilm terminals, 9e

better user measurement and analysis, 9f

more work in publication quality hardcopy output, 9g

Re (23105,) Implementation for Filtered Copy Plex, etc,

Re (23105,) Implementation for Filtered Copy Plex, etc, 1

I prefer method 2, so that I could learn and remember the following simple rule, 1a

> "If the filter doesn't pass a statement, its substatements move up a level." 1a1

Also, that is the way I have understood for many months (from Dornbush way back then) that New NLS was going to work, and thus have I written the documentation for the new Copy with Filter command, 1b

If you do eventually decide to stick with method 1 ("If the filter doesn't pass a statement, its first substatement is moved up into its place, with following substatements remaining at their old level, now having the first one as their source,"??!) or one other than 2, please notify me so that I can change the documentation. Thank you, Jeanne B, 1c

RLL 25-MAY-74 11:29 23120

Return file stack (ring) bug!!

Guess you al know that the file return stack (ring?) does not work as advertized. If one loads three or more files, thenproceeds to do a J[ump] ,fr (in TNLS) the previous file is loaded, another jump ,fr brings back the last one; from then on one flips back and forth between only the last two files. As i understood it, one could do repeated ,fr and eventually return to the very first one. Now one must count the position in the stack in order to return. Bad. 1

RLL 25-MAY-74 11:56 23121

CM not repositioned after deltion of last charcter in statement.

If one attempts to insert text at end of statement after last charcter or word has been deleted the system response with 'fst entry nonexistant'. Obviously the CM must be repositioned to point to the last character rather than the now non-existant last character. Try it. Also perhaps this message could be made a bit clearer. 1

KIRK 25-MAY-74 16:51 23122

Bug with ,fr: and ,r: in Jump to Link command

,fr: and ,r: (with colons) don't work in jump to link and they mess up your return ring. 1

JMB 25-MAY-74 17:29 23123

Useroptions Feedback commands in TNLS

Why does the Show Feedback command only show the feedback mode, when you can also set and Reset Length and Indenting in TNLS? It's confusing not to be able to find out what your status is if you've been playing with them. 1

My opinion is that
   length = ##
   indenting = ##
should print underneath
   mode = ----------
for all tnls users. (They should show the defaults if the user hasn't changed them.) 2

Please let me know your solution to this problem ASAP so I can document the Show Feedback command properly. 3

KIRK 26-MAY-74 13:10 23124

Bug that stops work with Split screens and character size 0

I had a vertical split with character size 0 on the right. I tried to move the boundary a little towards the left and got stradda JSYS error. The split screen went away as if I had moved the boundary to the margin. I tried to split again and I got PUSHDOWN OVERFLOW and popped into exec.

1

JMB 26-MAY-74 14:55 23125

Need for restructuring or renaming in help database

I can't get to the syntax statements for Sendmail commands, i.e, the link <syntax !sendmail !authors> bombs out at AUTHORS?, and in Help
Show syntax sendmail authors
goes to the syntax for Editor's Insert Journal command. Do you see the search problem? It's too confusing to try and fix it in TNLS, so I wondered if you would look at it and do it? Thanks, Jeanne 1

JMB 26-MAY-74 15:31 23126

Jumping to Return in TNLS

It is very difficult and frustrating to Jump to Return in TNLS because there is no way of knowing if you have counted back the right number of moves to get what you want. There is no Show Return ring command as there is for the file return ring; and you can only get system feedback on where you'll be going AFTER the <CR> has executed the command (with a slash--Jump to A: ,3r /<CR>); you can't change your mind, like you can when DNLS feeds back the destination in the Jump to Return command, until you've already been moved, which has itself altered your return stack further so it's then even harder to figure out the correct number of jumps to give. 1

KIRK 26-MAY-74 16:03 23127

Insert Sendmail re-renamed

Someone changed the name delimiters on the Insert Sendmail Command, There is a note there saying not to, but it happened anyway so I unnamed it by placing a space in front of it. Syntax sendmail authors should work now. By the way, just, "syntax authors" also works as authors is not duplicated any where else. 1

JHB 27-MAY-74 10:34 23129

New NLS bug in dir listing

When doing Show Protection on individual file received msg, "reference to undefined interpreter value", rather than protect listing for the file. The dir name and file name were typed out and were accurate. 1

JHB 27-MAY-74 11:58 23130

New NLS sugestion

The system defaults to Item in the Jump to command, I recommend that it default to Statement in Break and Append, In that way experts would have one less keystroke, and users shifting from old NLS would not loose that convenience, 1

NDM 28-MAY-74 09:04 23131

Response to KEV's (23105,): Filtered Edits

My vote on KEV's four methods in (gjournal,23105,)

NDM 28-MAY-74 09:04 23131

Response to KEV's (23105,): Filtered Edits

My vote on KEV's (GJOURNAL,23105,). 1

Method Three seems the worst, since statement E gets affected by a change in a seperate branch (which adds to whatever confusion arises out of the other branch's change in structure). 2

Method Four makes little sense on the Delete with Filter, though I have no strong objections to it with the other commands. 3

> Retaining the structure could be of value in files formatted (say for COM) based on structure. But in the Delete command, the whole idea was to get rid of that statement. 3a

Method One would be fine, but a plex can be structurally broken and a relationship lost. A trade-off between import of subservience and plex relationships in structure. 4

Method Two seems most appropriate for Delete wih filter (a precedent of sorts was set with Append in moving everything up a level). I would also be satisfied with Two in the Copy and Move command. I am not sure how Transpose will be used. 5

DVN 28-MAY-74 10:03 23132

Quarterly Management Prod

I hate to be this way but you still owe paragraphs of the quarterly mgement report (A separate issue from the final report). Please check (hjournal,23063,) against your ident. 1

JHB 28-MAY-74 10:21 23133

User Development Report: Training Tour 1 - 11 May 74

This 4pp report describes the use of the new Basic TNLS Course in the NSW and Seismic environments. Comments welcome.

USER DEVELOPMENT REPORT: Training tour 1 May -- 11 May 74 1

Table of Contents 1a

SUMMARY OF MEETINGS/TRAINING,.....3 1b

Conclusions:.....3F 1b1

TRAINEES:.....4 1c

NSW (ARPA Monitor = Steve Crocker).....4A 1c1

SEISMIC (ARPA).....4B 1c2

COURSE DESIGN (Joint effort by RWW, JCN, MDK, DVN, CHI, JHB)...5 1d

COURSE CHANGES (see the revised Course Outline and Command....6 1e

Introduction.....6A 1e1

Addressing.....6B 1e2

Typing.....6C 1e3

Editing commands:.....6D 1e4

USER DEVELOPMENT REPORT: Training tour 1 May -- 11 May 74 2

This report includes a discussion of the course material covered, and lists the individuals trained under each project/client, their generic position, and course revisions, 2a

SUMMARY OF MEETINGS/TRAINING 3

The process of introducing NLS to these potentially key users has an interesting chronology. The NSW group met at the SRI Washington offices for two consecutive days of training. Five people were provided with three terminals, two TIs and a Delta Data. Although it would have been ideal to have one terminal for each person, this worked out due to the cooperativeness of the trainees. The course was designed to be covered in two full days and was aimed at the lowest common denominator that we could expect to have in our course: the person totally inexperienced with NLS. Of primary interest was the viability of an approach that did not include the hierarchical structure or viewspecs, and contained the minimal command set necessary to handle units of text. 3a

The low key approach, without any attempt to sell NLS, worked

well. Individuals were entering text as per the course outline within the first half hour, and progressed rapidly after that. Many questions arose about additional capabilities, but the response, "we'll get to that after this course", seemed to be quite acceptable. An increasing pressure to move to additional capabilties beyond the basic course developed, and seemed to be a motivational factor contributing to the course completion early the second day. At that point everyone seemed ready to agree that NLS was more than a text editor and they were anxious to learn additional commands. I added the editing command repetoire beyond the four given in the basic course with an emphasis an the intuitive syntactical relationships, viewing and structure. The introduction of these went well, and seemed to illustrate that indeed NLS is intuitive after a good start (or foundation) is made through a formal course. 3b

I've never seen such an enthusiastic, positive response to NLS or an NLS course. There were a number of suggestions and changes in the basic course as noted below. As a result, the Basic Course (Journal,22858,1) should be viable and ready for general usage by anyone who ends up in a teaching role. 3c

The following week in Boston gave us further experience with this teaching method. Circumstances were less than optimal because we did not have a terminal for all attendees and the course had to be given in their environment with the uusual distractions. The approach defaulted to a general presentation of NLS to a room full of people, most of whom were not going to persue it further at that time. This is one of those hair raising situations that DCE has dealt with over the years -- trying to communicate to novices what NLS really is. The concepts that comprise the more advanced courses (see -- bair, course,1) were used as an outline, and each concept was discussed in general terms. It was very difficult to maintain a low profile -- NLS almost requires superlatives to differentiate it from other textual processing systems. Nevertheless, after a 2 - 3 hr presentation with help from JCN and NDM, the meeting ajourned until the actual course the following day. 3d

During that time more emphasis was placed on understanding rather than actually doing, due to time and terminal contraints. By the end of the third day, the same material as for the NSW people had been covered, but not at the same skill level due to the restricted availability of online experience to the trainees. The architect, Bob Sheppard, felt he was prepared to teach others in the office, particularly those who had just observed the course. 3e

Conclusions: 3f

Utilizing a low key, basic formal course, as a beginning point for all new users was strongly supported. The changes in the course listed herein are important and should ensure the viabiltiy of this approach. It was also supported, as in previous experiences, that each trainee should have his own terminal during the course, and that the course should be held away from the daily exigincies of the user's working environment, preferably at another location. The need for a user guide was also supported, one that would be intuitively organized. 3f1

TRAINEES: 4

NSW (ARPA Monitor = Steve Crocker) 4a

Carlson, W E (LT, Bill) (202) 695-9149, 4a1

AFDSC/GMT, Air Force Data Services Center, Pentagon, Wash, DC 20330 (Computer Techniques) 4a1a

Weeks, D R (LT, Doug) 4a2

Finney, E F (Betty), secretary plus (202) 695-9147-9, 4a2a

Riddle, E A (Liz), secretary plus (202) 695-9147-9 4a2b

Crain L A (LT, Larry) (205) 279-4444 4a3

Simulation and Analysis Branch (SYOA), Air Force Sytems Design Center (AFDSDC), Gunter AFB, Alabama 36114 (Host: USC-ISI) 4a3a

SEISMIC (ARPA) 4b

Lacoss, R T (Dick) (617)) 253-7858 (pI for LLANTS) 4b1

Sheppard, R M (Bob) 2 (617) 253-7856 (ARCH) 4b1a

Lincoln Labs Seismic Group, 42 Carlton St, Cambridge, Mass 02142 4b1a1

Observers: 4b1b

O'brien, Mary, Data Librarian 4b1b1

Williams, Lorenzo, Typist 4b1b2

COURSE DESIGN (Joint effort by RWW, JCN, MDK, DVN, CHI, JHB) 5

JHB 29-APR-74, Draft of Basic TNLS Course (NSW)
Location: (MJOURNAL, 22856, 1:w) 5a

Comments: This is a draft of the course to be used in Wash, to introduce NLS with the primary goal of simplicity and ease of learning. The Wash DC people are connected with the ARPA/NSW project (S. Crocker). It considered experimental and all feedback is welcome. 5a1

COURSE CHANGES (See the revised Course Outline and Command Summary -- mjournal,22858,1:y) 6

Introduction 6a

The introduction of what NLS is, should not include the AKW concept or our global purposes to augment all knowledge workers. Instead, a list of the functional capabilities of the system may be briefly covered IF the trainees are receptive. If they are not, beginning work at the terminal should commence immediately to establish that the user can accomplish something rather easily, thus overcoming the skepticsm and threat that are commonly generated during initial encounters with NLS. 6a1

A separate, general introduction can afford the opportunity to determine the subsequent division of the course. After the half hour or so of overview, the trainer could conceivably have sufficient touch with a group of not more than ten trainees to select two groups that will progress at significantly different rates. To wit, those who have never experienced a terminal and are threatened by the unfamilarity of the NLS world vs. those who have used other computer systems in a time sharing, online environment. 6a2

The question of what should be covered in the introduction is best answered in process at this point, with the list of capabilities permitting the appropriate level of detail and depth (see -- Journal,22858,3c). 6a3

Addressing 6b

The address, "t" for tail, is necessary to enable a novice user to get to the end of his single level file quickly enabling him to continue to build or add to statements without printing the entire file. Here we see the real disadvantage of not having viewspecs or levels to facilitate abreviated views of the file. "e" could not be used because it would not work unless the CM were at ,0. 6b1

Typing 6c

Replace the third TNLS concept, "Printing", with "Typing." Printing is confused with the output processor ("runoff") functions in the advanced courses. 6c1

The slash (\) for "easy print" should be added. It appears to be easy to grasp, moreso than the address CR for defaulting to current marker location. 6c2

Editing commands: 6d

The command, insert text, and the intrastatement address for end of statement (>) must be added. The substitutute text in statement command originially thought to be adequate for all editing has a "new text" length limit of approximately 81 characters. The error message presented is less than intelligible and if the user continues to type, the system has a high probability of blowing up (pages of assembly-like error messages). 6d1

This will enable the user to add text to statements without the problems associated with substitute. The concept of a statement is very different, dependent upon the user's background, thus we need a wide latitude here. 6d1a

Copy statement is necesssary to allow interfile editing, particularly for extra-directory files. 6d2

Finis 7

CHI 28-MAY-74 10:37 23134

look into 23129

Ken, can you look into the interpreter variable problem described in 23129 ? 1

NDM 28-MAY-74 12:24 23135

Outline for User Programming section of Final Report

Not much here; shall I go on?

Outline for User Programming section of Final Report

User Programs System and Library 1

Introduction 1a

Given (13041,4die4), we have emphasized delivery to users of: 1a1

special purpose user programs, and 1a1a

the ability to program in L10 1a1b

Why these goals are of value 1a2

User Program Library 1b

User Programs 1b1

types, criterion for inclusion 1b1a

Access from NLS 1b2

Documentation 1b3

User L10 Programming 1c

Documentation 1c1

Interface to core procedures 1c2

List of User programs 1d

References 1e

MEJ 28-MAY-74 12:54 23136

Bon Voyage to ARC

27 May 1974

Dr. Douglas C. Engelbart
Director
Stanford Research Institute
Augmentation Research Center
333 Ravenswood Avenue
Menlo Park, California 94025

1

Dear Doug: 2

This will confirm the discussions between Jim Norton, you, and myself concerning my resignation from SRI-ARC, 30 June 1974: 3

1. The coming, 30 June 1974, cut-back in funding from ARPA for the general SRI-ARC project and the change in directions from a generalized "knowledge workshop" environment to one devoted more exclusively toward hardware and software development, leaves no financial support for the documentation and cataloging work I have been doing at ARC for the past four years. So far, no other sources of funding for this work have been apparent. 3a

2. The catalog database is, of course, far from completed, as we know, but according to our discussions I am bringing it to the best possible condition that I can in the remaining time I have left, preparatory to your taking it off line by 30 June. 3b

Jim Norton and I have agreed that the final input to the database will be during the week of June 1-7, leaving a little time before I must leave for us to prepare final proofing, printouts so that you will have a searchable hardcopy, and preparation of the database to go onto mag tapes and in ARCHIVE. 3b1

3. I think that by the middle to end of June we will have the database in as good condition as could be expected. As of this morning we have an additional 191 entries (plus the 495 input after the last catalog) to add to the database. I will have time to input most, at least, of the numbered hardcopy items, hopefully all. 3c

Therefore, from the looks of the present situation, I think we can meet the 30 June 1974 deadline of my departure with the database properly secured. If I have to leave a few days earlier (if my appointment comes through earlier to the new position I have applied

for), I still think the database can be secured, although perhaps not quite as much done on it. 4

Working at SRI-ARC has been a tremendous experience. I like the group and know that your ideals and thinking have had, and will have, a large and helpful impact on both technology and the world in general. In other words, I liked it here....and regret the economic necessity of my leaving. I will always think of ARC as a second home. 5

All my very best wishes to all of you! 6

Sincerely,

Mildred E. Jernigan
Technical Coordinator 7

SRL 28-MAY-74 13:13 23137

file for quarterly report

My file, (lee,fere,) can be used as a rough draft for a section on feedback in the quarterly report. I'll let you know when I have it finished. 1

DVN 28-MAY-74 14:45 23138

DDSI Revamps Code, Substitutes Messenger for Courier

Last Friday I discussed with Terry Koker of DDSI their readyness to do pending jobs, the JOVIAL Manual and the Format Library, He said he had gone over the code and he believed that he had found the source of the spacing problem around changes to slanted typeface (link). 1

He said futher that the code "is essentially undebuggable" and needed substantial revamping, He believed he could have that revamping done by today or tomorrow and be able to run our jobs, He will call us, 2

AS part of the revamping he wants to stop using the face "Courier" and replace it with "Messenger", 3

Let me explain that Courier and Messenger look 99% alike, Courrier is a stick font, that is it is made up on the CRT in straight lines, All the characters of Courier take up te same width, but they achieve a kind of psuedo proportional spacing by varying the amount of white space on each side of different characters, and so can justify lines, 3a

Messenger is a painted font, created on the CRT with strokes, All its characters have the same widthand the space between them does not very, 3b

Both faces look pretty much like what comes out of a typrwriter or a line printer; we have used Courier when we wanted that effect, 3c

Koker asserted they coud save up to 50% running tme by getting rid of courier, He wants simply to supply Messenger when our file asks for Courier, I was agreable since Messenge will do what we want, However we will have to make two adjustments: 3d

Rewrite our spacing tables so that when we call for Courier/Messenger letters go in the right space, The rewrite will not be difficult if DDSI gives us the right number for the table, 3d1

For the same reason any old documents that contain Courier will have to be reprocesssed through a version of the Output Processor that contains the updated tables befoe it can be reprinted, (We can't simply haul a COM file off tape and send it to DDSI again,) 3d2

DVN 28-MAY-74 15:07 23139

Final Report Oultine, JOVIAL Deadline, Reading Messages

Besides what I reported in the journal item about Messenger and Courier, Koker hinted strongly that we should lay a deadline on DDSI that he could use as a leaver to get more time (machine time?) to work, I talked wth DLS and am going to tell them we would like to finish the whole thing this fiscal year, 1

I got a message from Rita Jordan this morning complaining, to make a long story short, about how Eileen had been reading the mail, Eileen did not realize the problem it created for Rita, I think it is straightened out now, 2

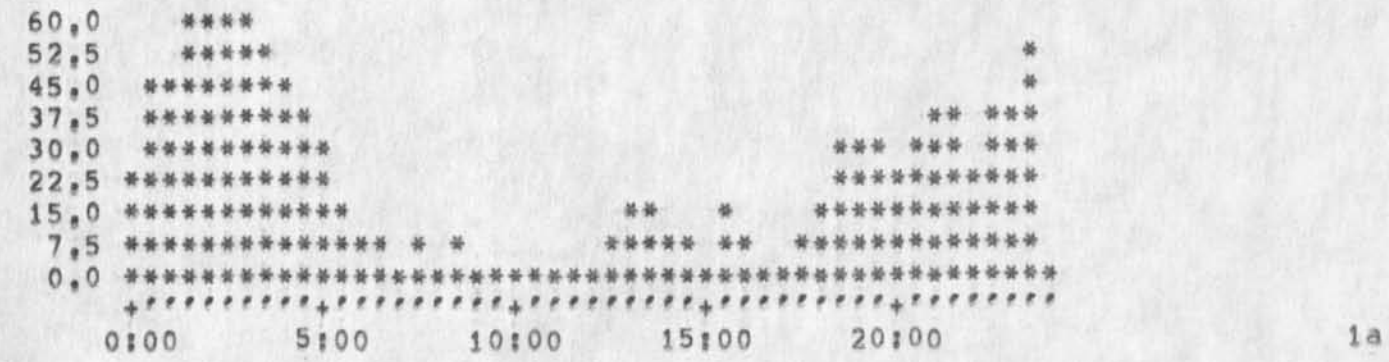
SRL 28-MAY-74 15:20 23140

Superwatch Average Graphs for Week of 5/12/74

It appears that percent used in DNLS is picking up only percent used in OLDNLS. I'll check on this and let you know what's happening. This is also true for the week 5/19/74.

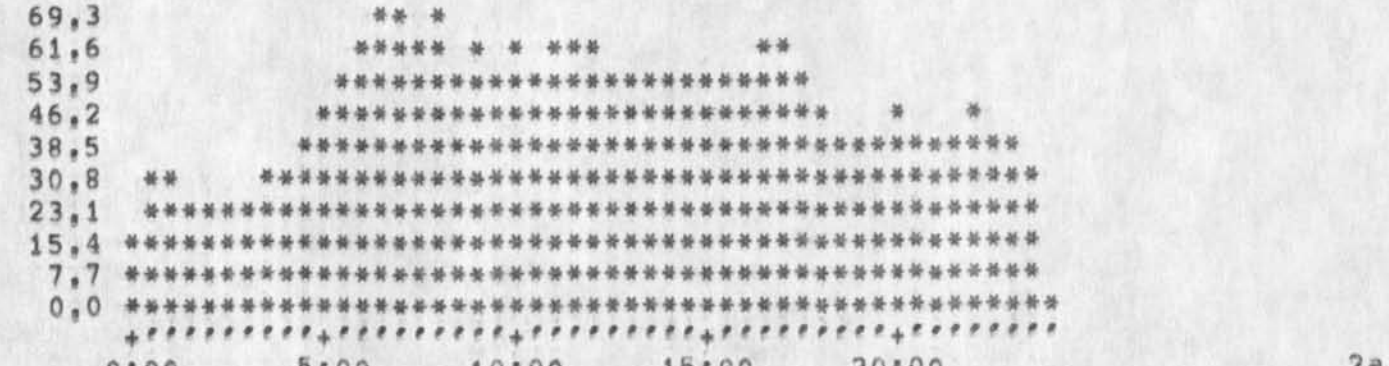
Superwatch Average Graphs for Week of 5/12/74

TIME PLOT OF AVERAGE IDLE TIME FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 1



1a

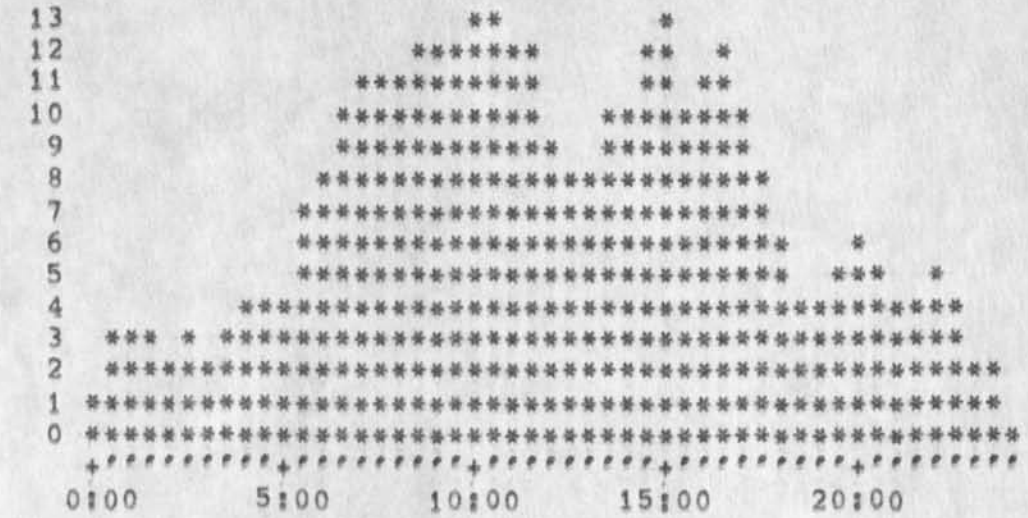
TIME PLOT OF AVERAGE PER CENT OF CPU TIME CHARGED TO USER ACCOUNTS
FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 2



2a

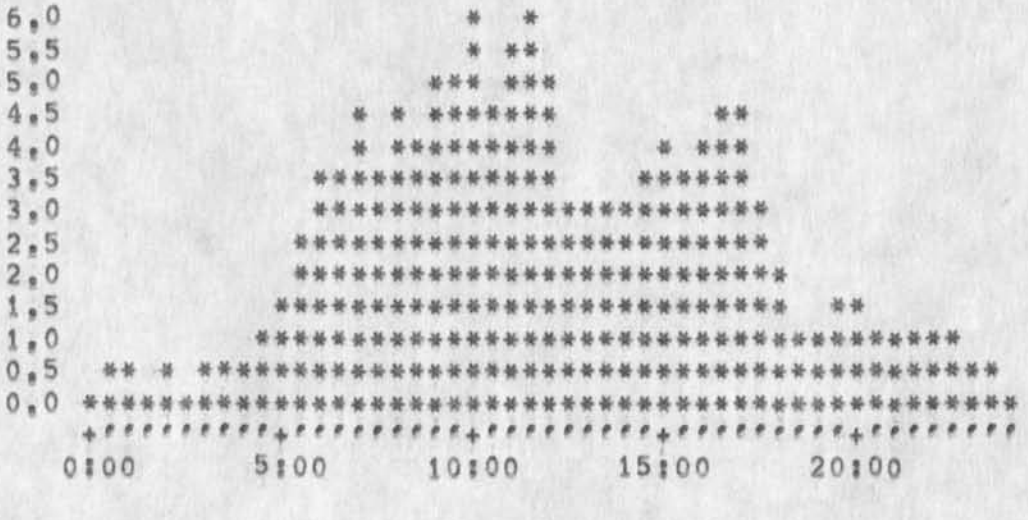
Superwatch Average Graphs for Week of 5/12/74

TIME PLOT OF AVERAGE NUMBER OF USERS FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 3



3a

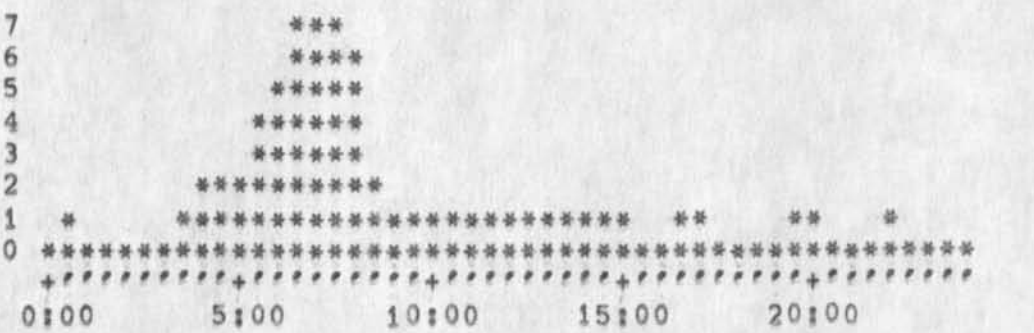
TIME PLOT OF AVERAGE NUMBER OF GO JOBS FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 4



4a

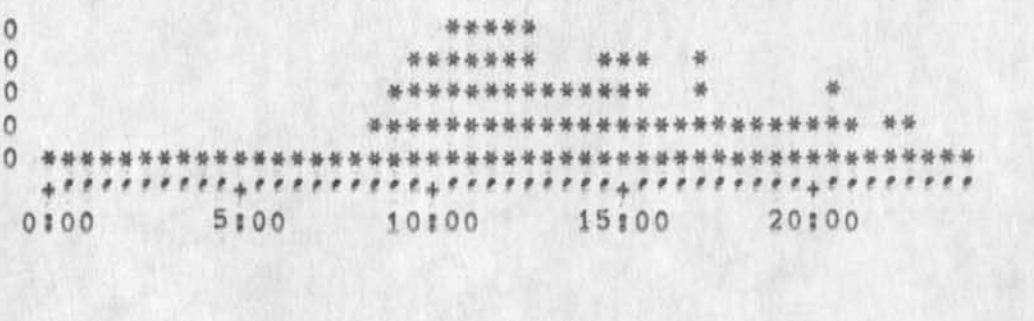
Superwatch Average Graphs for Week of 5/12/74

TIME PLOT OF AVERAGE NUMBER OF NETWORK USERS FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 5



5a

TIME PLOT OF AVERAGE PER CENT OF SYSTEM USED IN DNLS FOR WEEK OF 5/12/74
x axis labeled in units of hr:min, xunit = 30 minutes 6



6a

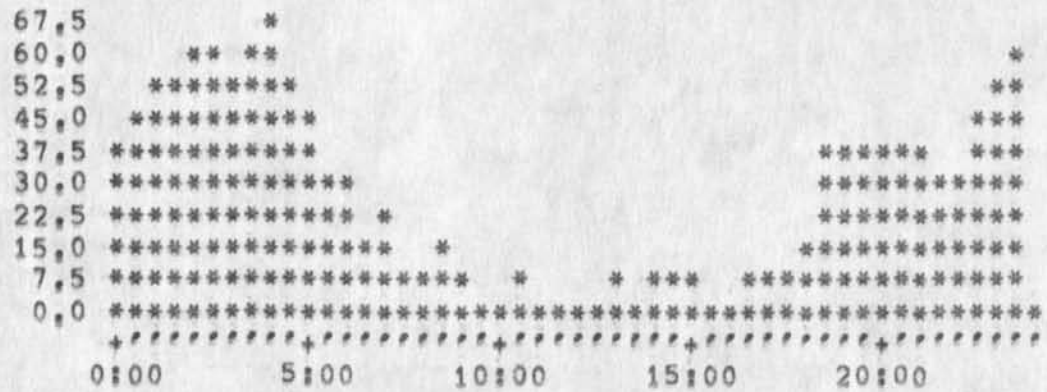
SRL 28-MAY-74 15:28 23141

Superwatch Average Graphs for Week of 5/19/74

For the weeks 5/5, 5/12, and 5/19, the graphs which say percent used in DNLS are in fact percent used in OLDDNLS - an interesting graph after all, I think it's significant that this % has steadily dropped over the 3 week period.

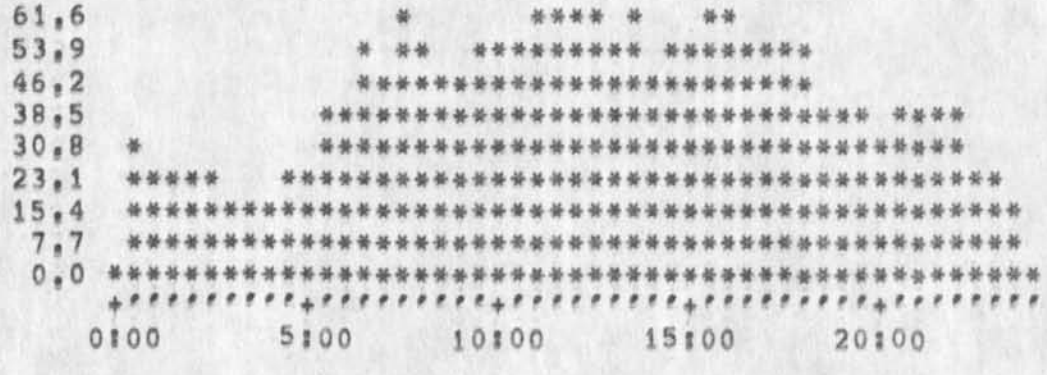
Superwatch Average Graphs for Week of 5/19/74

TIME PLOT OF AVERAGE IDLE TIME FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 1



1a

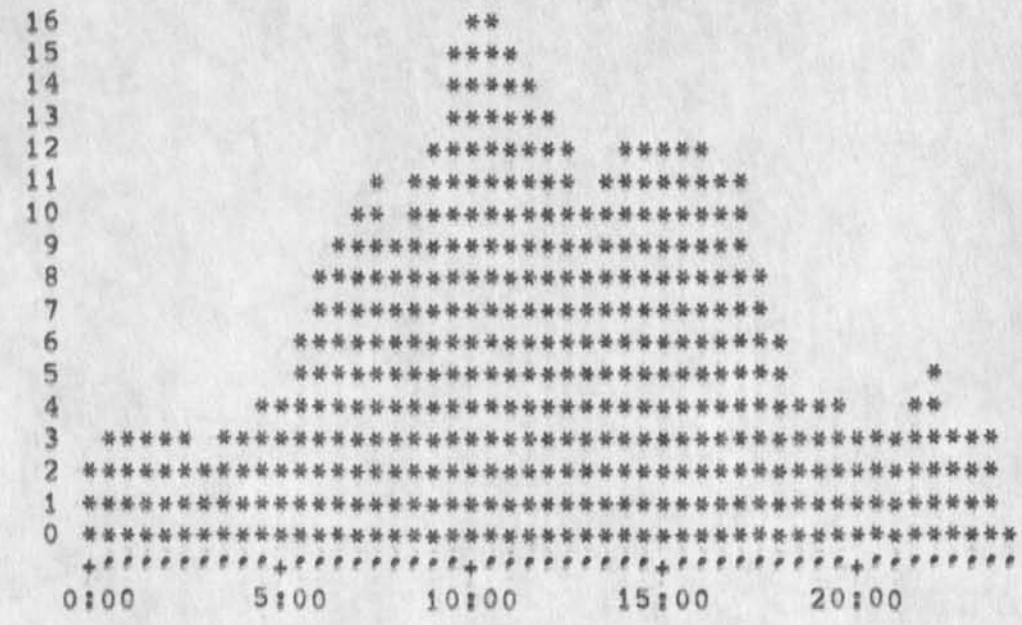
TIME PLOT OF AVERAGE PER CENT OF CPU TIME CHARGED TO USER ACCOUNTS
FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 2



2a

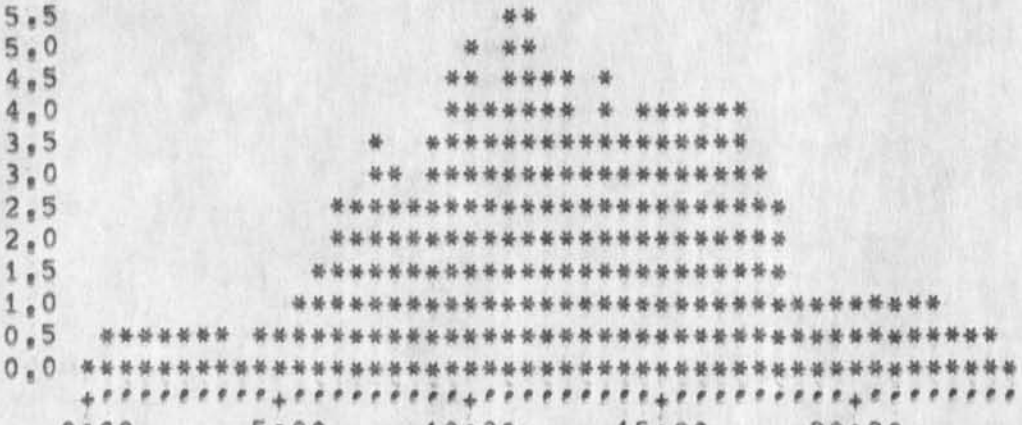
Superwatch Average Graphs for Week of 5/19/74

TIME PLOT OF AVERAGE NUMBER OF USERS FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 3



3a

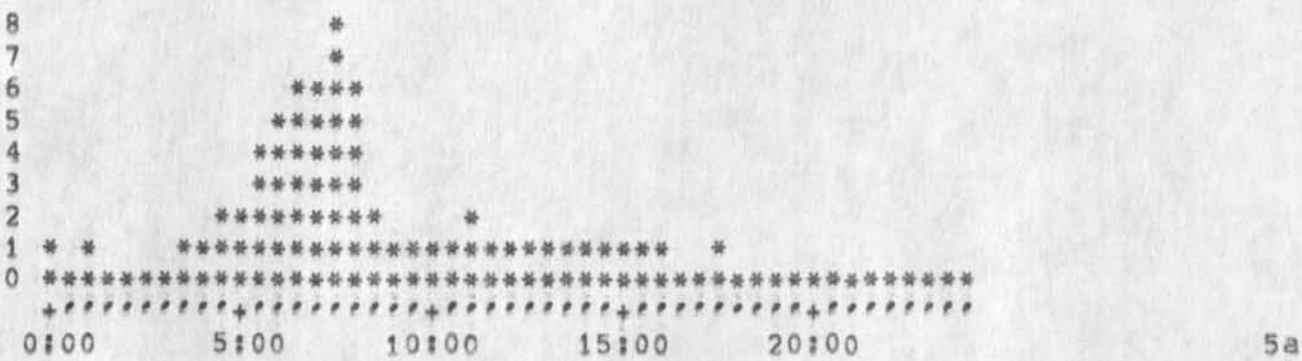
TIME PLOT OF AVERAGE NUMBER OF GO JOBS FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 4



4a

Superwatch Average Graphs for Week of 5/19/74

TIME PLOT OF AVERAGE NUMBER OF NETWORK USERS FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 5



5a

TIME PLOT OF AVERAGE PER CENT OF SYSTEM USED IN DNLS FOR WEEK OF 5/19/74
x axis labeled in units of hr:min, xunit = 30 minutes 6

6a

Miscellany for FEEDBACK

Load File BUGWORD VSPEC CA is what I'd still like to see, with extraction of a valid file name from the bugged visible, and with display of the deduced-file name before VSPEC CA (at least before the CA part). 1

Jump File BUG VSPEC CA is o.k., where assume you bug a link and it extracts file name -- but if Load File worked right, would rarely really need this option. Also, should show the file it will take you to before you commit yourself with the final CA. 2

Jump Link B;/A; VSPEC CA I'd also like, where the VSPEC would over-ride those provided in the link. 3

> In TNLS, I used to be able to say Print Branch (work, td; gebtzm) CA CA and have the VSPECS applied to the printout. Doesn't seem to work that way now? How come? Generally, I'd like to see the VSPEC of the last link used in an address apply to the viw (Print or Display), unless LIT vspecs added by user (which would over-ride conflicting link-held vspecs). 3a

Two Journal items with same number: (SENTMAIL 10 May 1455) 4

> Rcv'd May 09-1450 SRL: Request for More Feedback
> Sent: 9-MAY-74 14:15 (MJOURNAL, 22932, 1)
> Note: * action * 4a

> Rcv'd May 08-2033 JMB: Command Syntax Summary for New NLS--for online and offline viewing
> Sent: 4-MAY-74 21:38 (MJOURNAL, 22932,)
> Note: * action * 4b

Break Statement: As it was previously specified (Old NLS, and from OLD DCE) it is supposed to let you enter SPs that are inserted before the broken-off segment as it is made into a statement. Indeed, the '? menu shows that a SP is allowable after the bug; but a SP produces a "?" response, and I can't put leading SPs in front of the new statement. 5

> I seem to find from the ^Q HELP info that there isn't expected to be this provision, at least by the documenters. I want that provision back; in fact, I feel that one ought to be able to insert any charcters after an initial SP -- i.e. that all characters after the SP (that follows the BUG LEVADJ) and before the final CA should be inserted at the head of the breakoff statement. 5a

REPEATS: Nothing in the DNLS feedback that shows user that he is in Repeat mode -- e.g. finishing the literal for a new statement: if want to continue, may forget whether you are in Repeat, and don't

know whether to hit ^E or CA; or if want to stop inserting statements, don't know whether a CD or CA, 6

INTRODUCTION 1

This is a working paper I generated while designing a Multi-Host Journal System (MHJS). 1a

The MHJS is composed of program modules distributed appropriately through the Network. Each module implements a set of primitive operations which can be invoked by other modules in the System. 1a1

An initial list of modules and primitives was generated after considerable thought and several design meetings. 1a2

In this document I've 'implemented' each primitive in an imaginary, L10-like language to determine: whether or not the primitive is actually implementable, what parameters it must require, what data bases its module must maintain, what additional primitives need be defined, etc. 1a3

This document is UNFINISHED. Since ARC wil not receive funding for the MHJS, the incomplete design is offered as an aid to would-be implemeters of similar systems. There are a number of things which hadn't yet been included in this design. Among them are (off the top of my head): 1b

HOST FAILURES AND RESTART PROVISIONS 1b1

In this document, the assumption has been made that any inter-module call may fail because the addressed host is dead. Such failures are assumed to generate a signal which causes the calling primitive to fail and the error signal passed up the call stack. 1b1a

There may be cases where the primitive can be safely queued until the dead host is revived, but this approach hasn't been thought through. 1b1a1

Some modules (notably, the Registrar) require a restart mechanism by which, after a system crash and some arbitrary down period, they can effectively ask of another instance of the same module, "Say, what's happened since such-and-such a time and date when I went down?" 1b1b

LOCK MECHANISM 1b2

The Registrar turns out to be the module where most (if not all) race conditions are resolved, e.g., attempts to create two users or documents with the same name, or to publish the same document twice in a single Journal. Lock primitives

have been defined and are invoked by all primitives at the appropriate points in the pseudo-code, but the lock primitives themselves have not been fully coded, nor the means for detecting potential deadlocks thought out. 1b2a

LIMITING THE SIZE OF REGISTRAR DATA BASES 1b3

It turns out that a large number of data bases within the System have the following characteristics: 1b3a

The data base itself is a list of user names. 1b3a1

Primitives are required to permit the addition of names to, deletion of names from, replacement of names in, and retrieval of the contents of the list, all under appropriate access control. 1b3a2

Examples of such data bases are access lists, subscriber lists, reader lists, etc. 1b3b

The NLS concept of group idents thus turns out to be central to the MHJS design, and so a module called the Registrar (much like, but more complicated than, the NLS Ident System) was created to manipulate just such data bases. 1b3c

Such a module was already required for a more conventional purpose, namely to maintain information about human users of the system (as in NLS) and processes (which the MHJS considers users, since they have names, passwords, locations, etc.). This need required that Registrars cooperate with one another to distribute among themselves information about users, effectively maintaining copies of a single data base at a variety of locations throughout the Net. 1b3d

This communication is necessary because, in general, each user must be known throughout the Network, and known conveniently (i.e., without having to consult a second host). 1b3d1

Because of the Registrar's dual (and heavy) usage, the size of its data bases, replicated throughout the Network, threatened to become impractical. Two new concepts which are not represented in this document, therefore, needed to be introduced: 1b3e

Archiving group idents onto tertiary storage. The

implementation of such a mechanism can be transparent to other modules. 1b3e1

Limiting a group ident's domain of definition (or 'sphere', as it's called in this document) to a subset of the System. This mechanism must be appropriately employed whenever a user name list is created. 1b3e2

ACCOUNTING 1b4

Although access controls are applied throughout the MHJS, no accounting provisions had as yet been included. 1b4a

This document is TERSE, and should be read in parallel with (23144,) -- a prose treatment of the design. 1c

It was intended that the present document when completed become a quite detailed and therefore presumably useful implementaton specification and guide. 1c1

MODULES and their PROCEDURE CALLS 2

Storage Manager 2a

FUNCTION: 2a1

A Storage Manager provides an interface to some physical storage medium or media. It stores files by pathname, and provides primitives by which its callers can create files, retrieve their contents, and delete them. 2a1a

The Storage Manager specifies the name space from which pathnames can be chosen, and callers must adhere to that specification. 2a1b

Storage Managers are used by the System to store the contents of documents. 2a1c

DATA STRUCTURES (Pseudo-L10 Globals): 2a2

```
(filentry) RECORD STRING                                    2a2a
   pathname, % pathname %                                   2a2a1
   creator, % creator %                                     2a2a2
   createdate, % date and time of creation %                2a2a3
   readdate, % date and time of last read %                 2a2a4
   contents; % contents of file %                           2a2a5
DECLARE numfils;                                            2a2b
DECLARE ARRAY fillst;                                       2a2c
```

PORTS: 2a3

```
CRT-FILE (pathname ,contents ,readusrs ,writusrs ,delusrs
,ctrlusrs ,*rqstr ! YES/err)                                    2a3a

   Requestor:  anyone with write access to the Storage
   Manager                                                      2a3a1

   Description:                                                 2a3a2

      This primitive creates a file whose pathname and
      contents are as specified by the user.                    2a3a2a

      Read, write, delete, and controlling access are
      assigned to the specified lists of users.                 2a3a2b

   Pseudo-L10 Implementation:                                   2a3a3

      % locals %                                                2a3a3a
        LOCAL STRING pfx ,filset ,rqnm;                         2a3a3a1
        POINTER u;                                              2a3a3a2
      % lock required users %                                   2a3a3b
        ON ANY SIGNAL GOTO exit1;                               2a3a3b1
        #LCK-USRS# (rqstr ! self,WRIT ,RETR) @ anyreg;          2a3a3b2
      % verify requestor's identity %                           2a3a3c
        #VER-IDNTTY-USRS# (rqstr ,self);                        2a3a3c1
      % verify write access to storage manager %                2a3a3d
        #VER-MEM-USRS# (self,WRIT ,rqstr ,self);                2a3a3d1
      % clean names %                                           2a3a3e
        #PRS-USRS# (rqstr ! rqnm);                              2a3a3e1
      % create shorthands %                                     2a3a3f
        pfx _ self.pathname;                                    2a3a3f1
        filset _ pfx,READ ! pfx,WRIT ! pfx,DEL ! pfx,CTRL;      2a3a3f2
      % lock file, check for duplicate %                        2a3a3g
        #LCK-USRS# (filset ,CRT);                               2a3a3g1
      % create access lists %                                   2a3a3h
        ON ANY SIGNAL                                           2a3a3h1
          BEGIN                                                 2a3a3h1a
          #DEL-USRS# (filset ,self) @ anyreg;                   2a3a3h1b
          (exit1): UNLOCK fillst;                               2a3a3h1c
          #UNLCK-USRS# (rqstr ! self,WRIT ! filset);            2a3a3h1d
          END;                                                  2a3a3h1e
        #CRT-USR# (pfx,CTRL ,ctrlusrs ,ALL ,pfx,CTRL
         ,pfx,CTRL ,self ,null ,anyreg ,self);                  2a3a3h2
        #CRT-USR# (pfx,READ ,readusrs ,ALL ,pfx,CTRL
         ,pfx,CTRL ,self ,null ,anyreg ,self);                  2a3a3h3
        #CRT-USR# (pfx,WRIT ,writusrs ,ALL ,pfx,CTRL
         ,pfx,CTRL ,self ,null ,anyreg ,self);                  2a3a3h4
```

```
    #CRT-USR# (pfx,DEL ,delusrs ,ALL ,pfx,CTRL
     ,pfx,CTRL ,self ,null ,anyreq ,self);                    2a3a3h5
  % store the file %                                           2a3a3i
    LOCK fillst;                                              2a3a3i1
    fillst [BUMP numfils] _ u _ ALLOC filentry;               2a3a3i2
    u,pathname _ pathname;                                    2a3a3i3
    u,creator _ rqnm;                                         2a3a3i4
    u,createdate _ DATETIME;                                  2a3a3i5
    u,readdate _ null;                                        2a3a3i6
    u,contents _ contents;                                    2a3a3i7
  % return %                                                  2a3a3j
    RESET SIGNAL;                                             2a3a3j1
    UNLOCK fillst;                                            2a3a3j2
    #UNLCK-USRS# (rqstr ! self,WRIT ! filset);                2a3a3j3
    RETURN (YES);                                             2a3a3j4
    END,                                                      2a3a3j5
```

```
RETR-FILE (pathname ,*rqstr : (YES ,contents) / err)                    2a3b

   Requestor:  anyone with read access to the file                      2a3b1

   Description:                                                         2a3b2

      This primitive retrieves the contents of the file
      whose pathname is specified.                                      2a3b2a

   Pseudo-L10 Implementation:                                           2a3b3

      % locals %                                                        2a3b3a
        LOCAL STRING pfx ,result;                                       2a3b3a1
        POINTER u;                                                      2a3b3a2
      % create shorthands %                                             2a3b3b
        pfx _ self.pathname;                                            2a3b3b1
      % lock required users %                                           2a3b3c
        ON ANY SIGNAL                                                   2a3b3c1
           BEGIN                                                        2a3b3c1a
           UNLOCK fillst;                                               2a3b3c1b
           #UNLCK-USRS# (rqstr ! pfx.READ) @ anyreg;                    2a3b3c1c
           END;                                                         2a3b3c1d
        #LCK-USRS# (rqstr ! pfx.READ ,RETR) @ anyreg;                   2a3b3c2
      % verify requestor's identity %                                   2a3b3d
        #VER-IDNTTY-USRS# (rqstr ,self);                                2a3b3d1
      % verify read access %                                            2a3b3e
        #VER-MEM-USRS# (pfx.READ ,rqstr ,self);                         2a3b3e1
      % retrieve contents of file %                                     2a3b3f
        LOCK fillst;                                                    2a3b3f1
        u _  fillst [FIND (pathname)];                                  2a3b3f2
        result _ u.contents;                                            2a3b3f3
      % update read date %                                              2a3b3g
        u.readdate _ DATETIME;                                          2a3b3g1
      % return %                                                        2a3b3h
        EXECUTE SIGNAL;                                                 2a3b3h1
        RETURN (YES ,result);                                           2a3b3h2
        END.                                                            2a3b3h3
```

```
DEL-FILE (pathname ,*rqstr : YES/err)                                    2a3c

   Requestor:  anyone with delete access to the file                     2a3c1

   Description:                                                          2a3c2

      This primitive deletes the file whose pathname is
      specified, making both the physical storage it
      occupied and the pathname available for reuse,                     2a3c2a

   Pseudo-L10 Implementation:                                            2a3c3

      % locals %                                                         2a3c3a
        LOCAL i;                                                        2a3c3a1
        LOCAL STRING pfx ,filset;                                       2a3c3a2
      % create shorthands %                                             2a3c3b
        pfx _ self,pathname;                                            2a3c3b1
        filset _ pfx,READ ! pfx,WRIT ! pfx,DEL ! pfx,CTRL;              2a3c3b2
      % lock required users %                                           2a3c3c
        ON ANY SIGNAL                                                   2a3c3c1
           BEGIN                                                        2a3c3c1a
           UNLOCK fillst;                                               2a3c3c1b
           #UNLCK-USRS# (rqstr ! filset) @ anyreg;                      2a3c3c1c
           END;                                                         2a3c3c1d
        #LCK-USRS# (rqstr ! pfx,DEL ,RETR) @ anyreg;                    2a3c3c2
      % verify requestor's identity %                                   2a3c3d
        #VER-IDNTTY-USRS# (rqstr ,self);                                2a3c3d1
      % verify delete access %                                          2a3c3e
        #VER-MEM-USRS# (pfx,DEL ,rqstr ,self);                          2a3c3e1
      % lock file %                                                     2a3c3f
        #LCK-USRS# (filset ,DEL);                                       2a3c3f1
      % delete access lists %                                          2a3c3g
        #DEL-USRS# (filset ,self);                                      2a3c3g1
      % delete the file %                                              2a3c3h
        LOCK fillst;                                                    2a3c3h1
        DEALLOC (fillst [i _ FIND (pathname)]);                         2a3c3h2
        fillst [i] _ fillst [numfils := numfils-1];                     2a3c3h3
      % return %                                                       2a3c3i
        EXECUTE SIGNAL;                                                 2a3c3i1
        RETURN (YES);                                                   2a3c3i2
        END.                                                            2a3c3i3
```

Recorder 2b

FUNCTION: 2b1

A Recorder provides the same basic services as a Storage Manager. In a sense, it pre-processes store, retrieve and delete requests to one or more Storage Managers with which it deals. By so doing, it provides the following additional services to its callers: 2b1a

(1) Allows them to address documents by Accession Numbers (chosen from a System-wide name space), rather than by pathnames (chosen from a name space specific to a Storage Manager). 2b1a1

A Recorder must therefore implement a mapping between ANs and the pathnames permitted by the Storage Manager(s) whose services it chooses to use. 2b1a1a

(2) Keeps track of references to a document by requiring a list of users with every retrieve request and updating a master list of all users who've retrieved the document. 2b1a2

(3) Maintains the access control information for a document, forcing retrieve and delete requests to be channeled through the Recorder. 2b1a3

It implements a fifth type of access called 'distribute'. Only users with distribute access to a document can publish or deliver it. 2b1a3a

A Recorder grants no one write access to a document. 2b1a3b

(4) Permits copies of a document (called images) to exist at several Recorders. 2b1a4

(5) Allows the image of a document to be stored with provision for automatic deletion at a specified date and time, or if a specified interval goes by without a request for it. 2b1a5

DATA STRUCTURES (Pseudo-L10 Globals): 2b2

```
(imgentry) RECORD STRING                                  2b2a
  an, % AN %                                              2b2a1
  creator, % creator %                                    2b2a2
  createdate, % date and time of creation %               2b2a3
```

```
    readdate; % date and time of last read %                    2b2a4
  DECLARE numimgs;                                              2b2b
  DECLARE ARRAY imglst;                                         2b2c

PORTS;                                                          2b3
```

```
CRT-DOC (an ,numven ,content ,readusrs ,distusrs ,delusrs
,ctrlusrs ,*rqstr [,date/interval] : YES/err)                    2b3a

   Requestor:  anyone with write access to the Recorder          2b3a1

   Description:                                                  2b3a2

      This primitive is the counterpart of a Storage
      Manager's CRT-FILE primitive.  It stores the document
      whose contents are specified and associates with it
      the specified AN.                                          2b3a2a

         The Recorder verifies that the AN was assigned to
         the requesting user by the specified Number Vendor,
                                                                 2b3a2a1
      Read, distribute, delete, and controlling access are
      assigned to the specified lists of users.                  2b3a2b

      The caller has the option of specifying an interval or
      date at which the document is to be automatically
      deleted by the Recorder.                                   2b3a2c

   Pseudo-L10 Implementation:                                    2b3a3

      % locals %                                                 2b3a3a
        LOCAL STRING docset;                                     2b3a3a1
      % lock required users %                                    2b3a3b
        ON ANY SIGNAL GOTO exit1;                                2b3a3b1
        #LCK-USRS# (rqstr ! self,WRIT ,RETR) @ anyreg;           2b3a3b2
      % verify requestor's identity %                            2b3a3c
        #VER-IDNTTY-USRS# (rqstr ,self);                         2b3a3c1
      % verify write access to recorder %                        2b3a3d
        #VER-MEM-USRS# (self,WRIT ,rqstr ,self);                 2b3a3d1
      % verify requestor's ownership of AN %                     2b3a3e
        ON ANY SIGNAL GOTO exit2;                                2b3a3e1
        #CHNG-OWN-NUMS# (an ,self ,rqstr) @ numven;               2b3a3e2
      % create shorthands %                                      2b3a3f
        docset _ an,READ ! an,DIST ! an,DEL ! an,CTRL !
         an,READERS ! an,RECORDERS ! an,PUBLISHERS !
         an,HOME;                                                2b3a3f1
      % lock document, check for duplicate %                     2b3a3g
        #LCK-USRS# (docset ,CRT) @ anyreg;                       2b3a3g1
      % create access and other lists for document %             2b3a3h
        ON ANY SIGNAL                                            2b3a3h1
          BEGIN                                                  2b3a3h1a
          #DEL-IMG# (an ,self) @ self;                           2b3a3h1b
          #DEL-USRS# (docset ,self) @ anyreg;                    2b3a3h1c
```

```
      (exit2): #CHNG-OWN-NUM# (an ,rqstr ,self) @
       numven;                                                  2b3a3h1d
      (exit1): #UNLCK-USRS# (rqstr ! self,WRIT !
       docset) @ anyreg;                                        2b3a3h1e
      END;                                                      2b3a3h1f
    #CRT-USR# (an,HOME ,self ,ALL ,an,HOME ,null
     ,an,HOME ,null ,self);                                     2b3a3h2
    #CRT-USR# (an,CTRL ,ctrlusrs ,ALL ,an,CTRL ,an,CTRL
     ,an,HOME ,null ,self);                                     2b3a3h3
    #CRT-USR# (an,DIST ,distusrs ,ALL ,an,CTRL ,an,CTRL
     ,an,HOME ,null ,self);                                     2b3a3h4
    #CRT-USR# (an,READ ,readusrs ,ALL ,an,CTRL ,an,CTRL
     ! an,DIST ,an,HOME ,null ,self);                          2b3a3h5
    #CRT-USR# (an,DEL ,delusrs ,ALL ,an,CTRL ,an,CTRL
     ,an,HOME ,null ,self);                                     2b3a3h6
    #CRT-USR# (an,READERS ,null ,ALL ,null ,RECORDERS
     ,an,HOME ,null ,self);                                     2b3a3h7
    #CRT-USR# (an,RECORDERS ,null ,ALL ,RECORDERS
     ,RECORDERS ,an,HOME ,null ,self);                          2b3a3h8
    #CRT-USR# (an,PUBLISHERS ,null ,ALL ,PUBLISHERS
     ,PUBLISHERS ,an,HOME ,null ,self);                         2b3a3h9
  % store image %                                               2b3a3i
    #CRT-IMG# (an ,self ,null ,rqstr) @ self;                   2b3a3i1
  % mark AN used %                                              2b3a3j
    #USE-NUMS# (an ,self) @ numVen;                             2b3a3j1
  % handle auto-delete %                                        2b3a3k
    IF date/interval THEN DO OUT-OF-LINE                        2b3a3k1
       BEGIN                                                    2b3a3k1a
       special stuff                                            2b3a3k1b
       END;                                                     2b3a3k1c
  % return %                                                    2b3a3l
    RESET SIGNAL;                                               2b3a3l1
    #UNLCK-USRS# (rqstr ! self,WRIT ! docset) @ anyreg;
                                                                2b3a3l2
    RETURN (YES);                                               2b3a3l3
    END.                                                        2b3a3l4
```

CRT-IMG (an ,delusrs ,ctrlusrs ,*rqstr [,date/interval] :
YES/err) 2b3b

Requestor: anyone with write access to the Recorder 2b3b1

Description: 2b3b2

This primitive creates an image of the document whose AN is specified. Delete and controlling access to the image are assigned to the specified lists of users. 2b3b2a

The caller has the option of specifying an interval or date at which the image is to be automatically deleted by the Recorder. 2b3b2b

Pseudo-L10 Implementation: 2b3b3

```
% locals %                                              2b3b3a
  LOCAL STRING pfx ,rqnm ,imgset ,people ,rec ,txt;     2b3b3a1
  POINTER u;                                            2b3b3a2
% lock required users %                                 2b3b3b
  ON ANY SIGNAL GOTO exit1;                             2b3b3b1
  #LCK-USRS# (rqstr ! self,WRIT ! an,RECORDERS ,RETR)
   @ anyreg;                                            2b3b3b2
% verify requestor's identity %                        2b3b3c
  #VER-IDNTTY-USRS# (rqstr ,self);                      2b3b3c1
% verify write access to recorder %                     2b3b3d
  #VER-MEM-USRS# (self,WRIT ,rqstr ,self);              2b3b3d1
% clean names %                                         2b3b3e
  #PRS-USRS# (rqstr : rqnm);                            2b3b3e1
% create shorthands %                                   2b3b3f
  pfx _ self,an;                                        2b3b3f1
  imgset _ pfx,DEL ! pfx,CTRL;                          2b3b3f2
% lock image, check for duplicate %                     2b3b3g
  #LCK-USRS# (imgset ,CRT);                             2b3b3g1
% locate image to copy %                                2b3b3h
  % fetch list of recorders %                           2b3b3h1
    #RETR-MEM-USR# (an,RECORDERS ,self : people);       2b3b3h1a
  % attempt retrieve at each one %                       2b3b3h2
    FOR EVERY rec IN people DO                          2b3b3h2a
      IF #RETR-IMG# (an ,self : txt) @ rec THEN         2b3b3h2a1
        BEGIN                                           2b3b3h2a1a
        % create access lists for image %               2b3b3h2a1b
          ON ANY SIGNAL                                 2b3b3h2a1b1
            BEGIN                                       2b3b3h2a1b1a
            #DEL-FILE# (f(an) ,self) @
             anystorman;                                2b3b3h2a1b1b
```

```
                    #DEL-USRS# (imgset ,self) @ anyreg;
                                                        2b3b3h2a1b1c
                    (exit1): UNLOCK imglst;             2b3b3h2a1b1d
                    #UNLCK-USRS# (rqstr ! self,WRIT !
                     an,RECORDERS ! imgset);            2b3b3h2a1b1e
                    END;                                2b3b3h2a1b1f
                  #CRT-USR# (pfx,CTRL ,ctrlusrs ,ALL
                   ,pfx,CTRL ,pfx,CTRL ,self ,null ,self)
                   @ anyreg;                            2b3b3h2a1b2
                  #CRT-USR# (pfx,DEL ,delusrs ,ALL
                   ,pfx,CTRL ,pfx,CTRL ,self ,null
                   ,self);                              2b3b3h2a1b3
                % store image %                          2b3b3h2a1c
                  #CRT-FILE# (f(an) ,txt ,self ,self
                   ,self ,null ,self) @ anystorman;     2b3b3h2a1c1
                  LOCK imglst;                          2b3b3h2a1c2
                  imglst [BUMP numimgs] _ u _ ALLOC
                   imgentry;                            2b3b3h2a1c3
                  u,an _ an;                            2b3b3h2a1c4
                  u,creator _ rqnm;                     2b3b3h2a1c5
                  u,createdate _ DATETIME;              2b3b3h2a1c6
                  u,readdate _ null;                    2b3b3h2a1c7
                  UNLOCK imglst;                        2b3b3h2a1c8
                % add self to recorder list %           2b3b3h2a1d
                  #CHNG-MEM-USR# (an,RECORDERS ,ADD ,self
                   ,self) @ anyreg;                     2b3b3h2a1d1
                % return %                              2b3b3h2a1e
                  RESET SIGNAL;                         2b3b3h2a1e1
                  #UNLCK-USRS# (rqstr ! self,WRIT !
                   an,RECORDERS ! imgset);              2b3b3h2a1e2
                  RETURN (YES);                         2b3b3h2a1e3
                  END;                                  2b3b3h2a1e4
% return unsuccessful %                                    2b3b31
  SIGNAL (err);                                           2b3b311
  END.                                                    2b3b312
```

```
RETR-IMG (an ,*rqstrs : (YES ,content) / err)                    2b3c

   Requestor:  another Recorder
                  -- or --
               anyone with read access to the document
                 and to the Recorder                             2b3c1

   Description:                                                  2b3c2

      This primitive is the counterpart of a Storage
      Manager's RETR-FILE primitive.  It returns to the
      caller the contents of the document whose AN is
      specified.                                                 2b3c2a

   Pseudo-L10 Implementation:                                    2b3c3

      % locals %                                                 2b3c3a
         LOCAL copy;                                             2b3c3a1
         LOCAL STRING result;                                    2b3c3a2
         POINTER u;                                              2b3c3a3
      % lock required users %                                    2b3c3b
         ON ANY SIGNAL                                           2b3c3b1
            BEGIN                                               2b3c3b1a
            UNLOCK imglst;                                      2b3c3b1b
            #UNLCK-USRS# (rqstrs ! RECORDERS ! self,READ !
             an,READ) @ anyreg;                                 2b3c3b1c
            END;                                                2b3c3b1d
         #LCK-USRS# (rqstrs ! RECORDERS ! self,READ !
          an,READ ,RETR) @ anyreg;                              2b3c3b2
      % verify requestor's identity %                           2b3c3c
         #VER-IDNTTY-USRS# (rqstrs ,self);                      2b3c3c1
      % verify read access %                                    2b3c3d
         IF #VER-MEM-USRS# (RECORDERS ,rqstrs ,self)            2b3c3d1
            THEN copy _ TRUE;                                   2b3c3d1a
         ELSE                                                   2b3c3d2
            BEGIN                                               2b3c3d2a
            #VER-MEM-USRS# (self,READ ,rqstrs ,self);           2b3c3d2b
            #VER-MEM-USRS# (an,READ ,rqstrs ,self);             2b3c3d2c
            copy _ FALSE;                                       2b3c3d2d
            END;                                                2b3c3d2e
      % retrieve contents of document %                         2b3c3e
         #RETR-FILE# (f(an) ,self ; result) @ anystorman;       2b3c3e1
      % update list of readers and read date %                  2b3c3f
         IF NOT copy THEN                                       2b3c3f1
            BEGIN                                               2b3c3f1a
            #CHNG-MEM-USR# (an,READERS ,ADD ,rqstrs ,self) @
             anyreg;                                            2b3c3f1b
```

```
        LOCK imglst;                                    2b3c3f1c
        u _ imglst [FIND (an)];                         2b3c3f1d
        u.readdate _ DATETIME;                          2b3c3f1e
        UNLOCK imglst;                                  2b3c3f1f
        END;                                            2b3c3f1g
% return %                                               2b3c3g
    EXECUTE SIGNAL;                                     2b3c3g1
    RETURN (YES ,result);                               2b3c3g2
    END.                                                2b3c3g3
```

```
DEL-DOC (an ,*rqstr : YES/err)                                    2b3d

   Requestor:  anyone with delete access to the document          2b3d1

   Description:                                                   2b3d2

      This primitive deletes the document whose AN is
      specified -- all of its images, and its reader,
      recorder, publisher, home, and access lists.
      Publishers of the document are notified of the
      deletion.                                                   2b3d2a

   Pseudo-L10 Implementation:                                     2b3d3

      % locals %                                                  2b3d3a
        LOCAL STRING docset ,people ,pub ,rec;                    2b3d3a1
      % create shorthands %                                       2b3d3b
        docset _ an,READ ! an,DIST ! an,DEL ! an,CTRL !
         an,READERS ! an,RECORDERS ! an,PUBLISHERS !
         an,HOME;                                                 2b3d3b1
      % lock required users %                                     2b3d3c
        ON ANY SIGNAL (exit1): #UNLCK-USRS# (rqstr !
         docset) @ anyreg;                                        2b3d3c1
        #LCK-USRS# (rqstr ! an,DEL ! an,HOME ,RETR) @
         anyreg;                                                  2b3d3c2
      % verify requestor's identity %                             2b3d3d
        #VER-IDNTTY-USRS# (rqstr ,self);                          2b3d3d1
      % verify delete access %                                    2b3d3e
        #VER-MEM-USRS# (an,DEL ,rqstr ,self);                     2b3d3e1
      % lock document %                                           2b3d3f
        #LCK-USRS# (docset ,DEL);                                 2b3d3f1
      % delete images %                                           2b3d3g
        #RETR-MEM-USR# (an,RECORDERS ,self : people);             2b3d3g1
        DISABLE SIGNALS;                                          2b3d3g2
        FOR EVERY rec IN people DO                                2b3d3g3
           #DEL-IMG# (an ,self) @ rec;                            2b3d3g3a
      % delete the document's access and other lists %            2b3d3h
        ON ANY SIGNAL GOTO exit1;                                 2b3d3h1
        #DEL-USRS# (docset ,self) @ anyreg;                       2b3d3h2
      % notify publishers %                                       2b3d3i
        #RETR-MEM-USR# (an,PUBLISHERS ,self : people);            2b3d3i1
        DISABLE SIGNALS;                                          2b3d3i2
        FOR EVERY pub IN people DO                                2b3d3i3
           #DEL-ART# (an ,self) @ pub;                            2b3d3i3a
      % return %                                                  2b3d3j
        EXECUTE SIGNAL;                                           2b3d3j1
        RETURN (YES);                                             2b3d3j2
```

```
END,                                                    2b3d3j3
```

```
DEL-IMG (an ,*rqstr ! YES/err)                                          2b3e

   Requestor:  another Recorder
                  -- or --
               anyone with delete access to the image                   2b3e1

   Description:                                                         2b3e2

      This primitive is the counterpart of a Storage
      Manager's DEL-FILE primitive,  It deletes the image of
      the document whose AN is specified.                               2b3e2a

   Pseudo-L10 Implementation:                                           2b3e3

      % locals %                                                        2b3e3a
        LOCAL i;                                                        2b3e3a1
        LOCAL STRING pfx ,imgset;                                       2b3e3a2
      % create shorthands %                                             2b3e3b
        pfx _ self,an;                                                  2b3e3b1
        imgset _ pfx,DEL ! pfx,CTRL;                                    2b3e3b2
      % lock required users %                                           2b3e3c
        ON ANY SIGNAL                                                   2b3e3c1
           BEGIN                                                        2b3e3c1a
           UNLOCK imglst;                                               2b3e3c1b
           #UNLCK-USRS# (rqstr ! RECORDERS ! imgset) @
            anyreg;                                                     2b3e3c1c
           END;                                                         2b3e3c1d
        #LCK-USRS# (rqstr ! RECORDERS ! pfx,DEL ,RETR) @
         anyreg;                                                        2b3e3c2
      % verify requestor's identity %                                   2b3e3d
        #VER-IDNTTY-USRS# (rqstr ,self);                                2b3e3d1
      % verify delete access to image %                                 2b3e3e
        #VER-MEM-USRS# (RECORDERS ! pfx,DEL ,rqstr ,self);              2b3e3e1
      % lock image %                                                    2b3e3f
        #LCK-USRS# (imgset ,DEL);                                       2b3e3f1
      % remove self from recorder list %                               2b3e3g
        #CHNG-MEM-USR# (an,RECORDERS ,DEL ,self ,self);                 2b3e3g1
      % delete image access lists %                                    2b3e3h
        #DEL-USRS# (imgset ,self);                                      2b3e3h1
      % delete the image %                                             2b3e3i
        LOCK imglst;                                                    2b3e3i1
        DEALLOC (imglst [i _ FIND (an)]);                               2b3e3i2
        imglst [i] _ imglst [numimgs := numimgs-1];                     2b3e3i3
        UNLOCK imglst;                                                  2b3e3i4
        #DEL-FILE# (f(an) ,self) @ anystorman;                          2b3e3i5
      % return %                                                       2b3e3j
        EXECUTE SIGNAL;                                                 2b3e3j1
```

```
RETURN (YES);                                          2b3e3j2
END.                                                   2b3e3j3
```

Publisher 2c

FUNCTION: 2c1

A Publisher distributes notices of recorded documents to its Subscribers. 2c1a

DATA STRUCTURES (Pseudo-L10 Globals): 2c2

DECLARE journaltype; 2c2a

PORTS: 2c3

```
CRT-ART (an ,citation ,delusrs ,ctrlusrs ,*rqstr : YES/err)     2c3a

   Requestor:  anyone with write access to the publisher
                 and distribute access to the document          2c3a1

      Also ...                                                  2c3a1a

         If the journal is PRIVATE:                             2c3a1a1
            requestor needs controlling access to the
             document                                           2c3a1a1a
              and the document must be unpublished              2c3a1a1b
         If the journal is CONTROLLED:                          2c3a1a2
            requestor needs controlling access to the
             document                                           2c3a1a2a
                          -- or --                              2c3a1a2b
            the document's delete access list must be null      2c3a1a2c
              and its controlling access list contain only
                PUBLISHERS                                      2c3a1a2d
         If the journal is UNCONTROLLED:                        2c3a1a3
            there are no additional requirements                2c3a1a3a

   Description:                                                 2c3a2

      This primitive causes the document whose AN is
      specified to be published in the journal.  The caller
      specifies the document's citation.  Delete and
      controlling access to the article are assigned to the
      specified lists of users.                                 2c3a2a

   Pseudo-L10 Implementation:                                   2c3a3

      % locals %                                                2c3a3a
         LOCAL STRING pfx ,artset ,sh ,people ,sub;             2c3a3a1
      % create shorthands %                                     2c3a3b
         pfx _ self,an;                                         2c3a3b1
         artset _ pfx,DEL ! pfx,CTRL;                           2c3a3b2
         sh _ rqstr ! self,WRIT ! an,DIST ! an,CTRL ! an,DEL
          ! self,SUBSCRIBERS ! self,READERS;                    2c3a3b3
      % lock required users %                                   2c3a3c
         ON ANY SIGNAL #UNLCK-USRS# (sh ! artset) @ anyreg;     2c3a3c1
         #LCK-USRS# (sh ,RETR) @ anyreg;                        2c3a3c2
      % verify requestor's identity %                           2c3a3d
         #VER-IDNTTY-USRS# (rqstr ,self);                       2c3a3d1
      % verify write access to the journal %                    2c3a3e
         #VER-MEM-USRS# (self,WRIT ,rqstr ,self);               2c3a3e1
      % verify distribute access %                              2c3a3f
         #VER-MEM-USRS# (an,DIST ,rqstr ,self);                 2c3a3f1
```

```
% lock article, check for duplicate %                          2c3a3g
   #LCK-USRS# (artset ,CRT);                                   2c3a3g1
% check journal-type dependent access restrictions %           2c3a3h
   CASE journaltype OF                                         2c3a3h1
      = private:                                               2c3a3h1a
        BEGIN                                                  2c3a3h1a1
        % check for other publishers %                         2c3a3h1a2
           #RETR-MEM-USR# (an,PUBLISHERS ,self :
            people);                                           2c3a3h1a2a
           IF people # null THEN SIGNAL (err);                 2c3a3h1a2b
        % verify controlling access %                          2c3a3h1a3
           #CHNG-MEM-USR# (an,CTRL ,REPL ,self
            ,rqstr);                                           2c3a3h1a3a
        % change other access lists %                          2c3a3h1a4
           #CHNG-MEM-USR# (an,READ ,REPL
            ,self,READERS ,self);                              2c3a3h1a4a
           #CHNG-MEM-USR# (an,DIST ,REPL ,null
            ,self);                                            2c3a3h1a4b
           #CHNG-MEM-USR# (an,DEL ,REPL ,null ,self);
                                                               2c3a3h1a4c
        END;                                                   2c3a3h1a5
      = controlled:                                            2c3a3h1b
        BEGIN                                                  2c3a3h1b1
        % verify optional controlling access %                 2c3a3h1b2
           IF #CHNG-MEM-USR# (an,CTRL ,REPL
            ,PUBLISHERS ,rqstr) THEN                           2c3a3h1b2a
              #CHNG-MEM-USR# (an,DEL ,REPL ,null
               ,self)                                          2c3a3h1b2a1
        % verify document previously published %               2c3a3h1b3
           ELSE                                                2c3a3h1b3a
              BEGIN #RETR-MEM-USR# (an,CTRL ,self :
               people);                                        2c3a3h1b3a1
              IF people # PUBLISHERS THEN SIGNAL
               (err);                                          2c3a3h1b3a2
              #RETR-MEM-USR# (an,DEL ,self : people);
                                                               2c3a3h1b3a3
              IF people # null THEN SIGNAL (err);
                                                               2c3a3h1b3a4
              END;                                             2c3a3h1b3a5
        % give subscribers' readers read access %              2c3a3h1b4
           #CHNG-MEM-USR# (an,READ ,ADD ,self,READERS
            ,self);                                            2c3a3h1b4a
        END;                                                   2c3a3h1b5
      = uncontrolled:                                          2c3a3h1c
        % give subscribers' readers read access %              2c3a3h1c1
           #CHNG-MEM-USR# (an,READ ,ADD ,self,READERS
            ,rqstr);                                           2c3a3h1c1a
```

```
% create access lists %                                          2c3a3i
   #CRT-USR# (pfx,CTRL ,ctrlusrs ,ALL ,pfx,CTRL
    ,pfx,CTRL ,self ,null ,self);                                2c3a3i1
   #CRT-USR# (pfx,DEL ,delusrs ,ALL ,pfx,CTRL
    ,pfx,CTRL ,self ,null ,self);                                2c3a3i2
% add document to master catalog %                               2c3a3j
   #ADD-DOC-CAT# (an ,citation ,rqstr ,self) @ mascat;
                                                                 2c3a3j1
% update publisher list %                                        2c3a3k
   #CHNG-MEM-USR# (an,PUBLISHERS ,ADD ,self ,self) @
    anyreg;                                                      2c3a3k1
% notify subscribers %                                          2c3a31
   #RETR-MEM-USR# (self,SUBSCRIBERS ,self : people) @
    anyreg;                                                      2c3a311
   DISABLE SIGNALS;                                              2c3a312
   FOR EVERY sub IN people DO                                    2c3a313
      #ADD-DOC-CAT# (an ,citation ,rqstr ,self) @ sub;
                                                                 2c3a313a
% return %                                                       2c3a3m
   EXECUTE SIGNAL;                                               2c3a3m1
   RETURN (YES);                                                 2c3a3m2
   END,                                                          2c3a3m3
```

```
DEL-ART (an ,*rqstr : YES/err)                                        2c3b

   Requestor:  another Recorder
                  -- or --
               anyone with delete access to the article               2c3b1

   Description:                                                       2c3b2

      This primitive causes the article whose AN is
      specified to be deleted from the journal.                       2c3b2a

   Pseudo-L10 Implementation:                                         2c3b3

      % locals %                                                      2c3b3a
        LOCAL STRING sh ,pfx ,artset ,people ,sub;                    2c3b3a1
      % refuse unless journal uncontrolled %                          2c3b3b
        IF journaltype # uncontrolled THEN SIGNAL (err);              2c3b3b1
      % create shorthands %                                           2c3b3c
        pfx _ self.an;                                                2c3b3c1
        artset _ pfx.DEL ! pfx.CTRL;                                  2c3b3c2
        sh _ rqstr ! RECORDERS ! pfx.DEL !
         self.SUBSCRIBERS;                                            2c3b3c3
      % lock required users %                                         2c3b3d
        ON ANY SIGNAL #UNLCK-USRS# (sh ! artset) @ anyreq;            2c3b3d1
        #LCK-USRS# (sh ,RETR) @ anyreq;                               2c3b3d2
      % verify requestor's identity %                                 2c3b3e
        #VER-IDNTTY-USRS# (rqstr ,self);                              2c3b3e1
      % verify delete access %                                        2c3b3f
        #VER-MEM-USRS# (RECORDERS ! pfx.DEL ,rqstr ,self);            2c3b3f1
      % lock article %                                                2c3b3g
        #LCK-USRS# (artset ,DEL);                                     2c3b3g1
      % delete self from list of publishers %                         2c3b3h
        #CHNG-MEM-USR# (an.PUBLISHERS ,DEL ,self ,self);              2c3b3h1
      % delete document from master catalog %                         2c3b3i
        #DEL-DOC-CAT# (an ,self) @ mascat;                            2c3b3i1
      % delete access lists %                                         2c3b3j
        #DEL-USRS# (artset ,self) @ anyreq;                           2c3b3j1
      % notify subscribers %                                          2c3b3k
        #RETR-MEM-USR# (self.SUBSCRIBERS ,self : people);             2c3b3k1
        DISABLE SIGNALS;                                              2c3b3k2
        FOR EVERY sub IN people DO                                    2c3b3k3
          #DEL-DOC-CAT# (an ,self) @ sub;                             2c3b3k3a
      % return %                                                      2c3b3l
        EXECUTE SIGNAL;                                               2c3b3l1
        RETURN (YES);                                                 2c3b3l2
        END.                                                          2c3b3l3
```

Cataloger 2d

FUNCTION: 2d1

A Cataloger maintains a data base containing information about each document in a Publisher's journal, and provides a mechanism by which the data base can be interrogated by users. 2d1a

Each Publisher and each of its Subscribers has a Cataloger. 2d1a1

DATA STRUCTURES (Pseudo-L10 Globals): 2d2

```
(catentry) RECORD STRING                                  2d2a
   an, % AN %                                             2d2a1
   contributor, % contributor %                           2d2a2
   contributedate, % date of contribution %               2d2a3
   readdate, % date of last read %                        2d2a4
   citation; % citation %                                 2d2a5
DECLARE numcats;                                          2d2b
DECLARE ARRAY catlst;                                     2d2c
```

PORTS: 2d3

```
ADD-DOC-CAT (an ,citation ,cntrbtr ,*rqstr : YES/err)                    2d3a

   Requestor:  anyone with write access to the Cataloger                 2d3a1

   Description:                                                          2d3a2

      This primitive adds to the catalog the newly-published
      document whose AN is specified.  The caller provides a
      citation.                                                          2d3a2a

   Pseudo-L10 Implementation:                                            2d3a3

      % locals %                                                         2d3a3a
         POINTER u;                                                      2d3a3a1
      % lock required users %                                            2d3a3b
         ON ANY SIGNAL                                                  2d3a3b1
            BEGIN                                                       2d3a3b1a
            UNLOCK catlst;                                              2d3a3b1b
            #UNLCK-USRS# (rqstr ! self,WRIT) @ anyreg;                  2d3a3b1c
            END;                                                        2d3a3b1d
         #LCK-USRS# (rqstr ! self,WRIT ,RETR) @ anyreg;                 2d3a3b2
      % verify requestor's identity %                                   2d3a3c
         #VER-IDNTTY-USRS# (rqstr ,self);                               2d3a3c1
      % verify write access %                                           2d3a3d
         #VER-MEM-USRS# (self,WRIT ,rqstr ,self);                       2d3a3d1
      % check for duplicate entry %                                     2d3a3e
         LOCK catlst;                                                   2d3a3e1
         FIND (an);                                                     2d3a3e2
      % add document to catalog %                                       2d3a3f
         catlst [BUMP numcats] _ u _ ALLOC catentry;                    2d3a3f1
         u,an _ an;                                                     2d3a3f2
         u,contributor _ cntrbtr;                                       2d3a3f3
         u,contributedate _ DATETIME;                                   2d3a3f4
         u,citation _ citation;                                         2d3a3f5
      % return %                                                        2d3a3g
         EXECUTE SIGNAL;                                                2d3a3g1
         RETURN (YES);                                                  2d3a3g2
         END.                                                           2d3a3g3
```

```
RETR-DOC-CAT (an ,#rqstr : (YES ,citation) / err)                    2d3b

   Requestor:  anyone with read access to the Cataloger              2d3b1

   Description:                                                      2d3b2

      This primitive returns the citation for the document
      whose AN is specified.                                         2d3b2a

   Pseudo-L10 Implementation:                                        2d3b3

      % locals %                                                     2d3b3a
        LOCAL result;                                               2d3b3a1
        POINTER u;                                                  2d3b3a2
      % lock required users %                                       2d3b3b
        ON ANY SIGNAL                                               2d3b3b1
           BEGIN                                                    2d3b3b1a
           UNLOCK catlst;                                           2d3b3b1b
           #UNLCK-USRS# (rqstr ! self,READ) @ anyreq;               2d3b3b1c
           END;                                                     2d3b3b1d
        #LCK-USRS# (rqstr ! self,READ ,RETR) @ anyreq;              2d3b3b2
      % verify's requestor's identity %                             2d3b3c
        #VER-IDNTTY-USRS# (rqstr ,self);                            2d3b3c1
      % verify read access %                                        2d3b3d
        #VER-MEM-USRS# (self,READ ,rqstr ,self);                    2d3b3d1
      % fetch citation and update read date %                       2d3b3e
        LOCK catlst;                                                2d3b3e1
        u _ catlst [FIND (an)];                                     2d3b3e2
        result _ u,citation;                                        2d3b3e3
        u,readdate _ DATETIME;                                      2d3b3e4
      % return %                                                    2d3b3f
        EXECUTE SIGNAL;                                             2d3b3f1
        RETURN (YES ,result);                                       2d3b3f2
        END.                                                        2d3b3f3
```

```
SRCH-CAT (key ,*rqstr : (YES ,ans) / err)                          2d3c

   Requestor:  anyone with read access to the Cataloger            2d3c1

   Description:                                                    2d3c2

      This primitive returns the ANs of the documents whose
      key is specified.  The key might be the contributor, a
      keyword in the citation, etc.                                2d3c2a

   Pseudo-L10 Implementation:                                      2d3c3

      % locals %                                                   2d3c3a
        LOCAL i;                                                   2d3c3a1
        LOCAL STRING an;                                           2d3c3a2
        POINTER u;                                                 2d3c3a3
      % lock required users %                                      2d3c3b
        ON ANY SIGNAL                                              2d3c3b1
           BEGIN                                                   2d3c3b1a
           UNLOCK catlst;                                          2d3c3b1b
           #UNLCK-USRS# (rqstr ! self,READ) @ anyreg;              2d3c3b1c
           END;                                                    2d3c3b1d
        #LCK-USRS# (rqstr ! self,READ ,RETR) @ anyreg;             2d3c3b2
      % verify requestor's identity %                              2d3c3c
        #VER-IDNTTY-USRS# (rqstr ,self);                           2d3c3c1
      % verify read access %                                       2d3c3d
        #VER-MEM-USRS# (self,READ ,rqstr ,self);                   2d3c3d1
      % determine documents sought %                               2d3c3e
        result _ null;                                             2d3c3e1
        LOCK catlst;                                               2d3c3e2
        FOR i _ 0 UNTIL numcats DO                                 2d3c3e3
           BEGIN                                                   2d3c3e3a
           u _ catlst [i];                                         2d3c3e3b
           IF TEST (u ,key) THEN result _ result ! an;             2d3c3e3c
           END;                                                    2d3c3e3d
      % return %                                                   2d3c3f
        EXECUTE SIGNAL;                                            2d3c3f1
        RETURN (YES ,result);                                      2d3c3f2
        END.                                                       2d3c3f3
```

```
RETR-CAT ,*rqstr : (YES ,catalog) / err)                                2d3d

   Requestor:  anyone with read access to the Cataloger                 2d3d1

   Description:                                                         2d3d2

      This primitive returns a copy of the entire catalog,
      The primitive can be employed to initialize one copy
      of a catalog from another (as a Subscriber's catalog
      must be initialized from the Publisher's), or to
      produce (for example) hardcopy of the catalog.                    2d3d2a

   Pseudo-L10 Implementation:                                           2d3d3

      % locals %                                                        2d3d3a
        LOCAL i;                                                       2d3d3a1
        LOCAL STRING result;                                           2d3d3a2
        POINTER u;                                                     2d3d3a3
      % lock required users %                                          2d3d3b
        ON ANY SIGNAL #UNLCK-USRS# (rqstr ! self,READ) @
         anyreg;                                                       2d3d3b1
        #LCK-USRS# (rqstr ! self,READ ,RETR) @ anyreg;                 2d3d3b2
      % verify requestor's identity %                                  2d3d3c
        #VER-IDNTTY-USRS# (rqstr ,self);                               2d3d3c1
      % verify read access %                                           2d3d3d
        #VER-MEM-USRS# (self,READ ,rqstr ,self);                       2d3d3d1
      % format the catalog for output %                                2d3d3e
        result _ null;                                                 2d3d3e1
        LOCK catlst;                                                   2d3d3e2
        FOR i _ 0 UNTIL numcats DO                                     2d3d3e3
          BEGIN                                                        2d3d3e3a
          u _ catlst [i];                                              2d3d3e3b
          result _ result ! u,an ! u,contributor !
           u,contributedate ! u,citation;                              2d3d3e3c
          END;                                                         2d3d3e3d
      % return %                                                       2d3d3f
        UNLOCK catlst;                                                 2d3d3f1
        EXECUTE SIGNAL;                                                2d3d3f2
        RETURN (YES ,result);                                          2d3d3f3
        END.                                                           2d3d3f4
```

```
Registrar                                                              2e

   FUNCTION;                                                           2e1

      A Registrar maintains information and responds to queries
      about users within the System.                                   2e1a

   DATA STRUCTURES (Pseudo-L10 Globals):                               2e2

      (userentry) RECORD STRING                                        2e2a
         name, % name %                                                2e2a1
         password, % password %                                        2e2a2
         creator, % creator %                                          2e2a3
         createdate, % date and time of creation %                     2e2a4
         readdate, % date and time of last read %                      2e2a5
         writedate, % date and time of last write %                    2e2a6
         usedate, % date and time of last use %                        2e2a7
         membership, % membership %                                    2e2a8
         affiliations, % users of which the user is a member %         2e2a9
         sphere, % registrars at which user is known %                 2e2a10
         ral, % users with read access to membership %                 2e2a11
         wal, % users with write access to membership %                2e2a12
         aal, % users with append access to membership %               2e2a13
         xal, % users with delete access to user %                     2e2a14
         cal, % users with controlling access to user %                2e2a15
         home; % home registrar %                                      2e2a16
      DECLARE numusrs;                                                 2e2b
      DECLARE ARRAY usrlst;                                            2e2c

   PORTS:                                                              2e3
```

```
CRT-USR (usr ,memusrs ,readusrs ,writusrs ,appusrs ,delusrs
,ctrlusrs ,sphere ,*rqstr [,crtusr ,crtdate] : YES/err)          2e3a

   Requestor:  another Registrar
                  -- or --
               anyone with write access to the Registrar          2e3a1

   Description:                                                   2e3a2

      This primitive creates a user with the specified name,
      password, and membership.  Read, write, append,
      delete, and controlling access are granted to the
      specified lists of users.                                   2e3a2a

      If the creating user is a Registrar, then the original
      creator and create date are specified.                      2e3a2b

   Pseudo-L10 Implementation:                                     2e3a3

      % locals %                                                   2e3a3a
         LOCAL slave;                                             2e3a3a1
         LOCAL STRING usrset ,memnms ,readnms ,writnms
          ,appnms ,delnms ,ctrlnms ,n ,rqnm ,usrnm ,sphnm
          ,crtnm ,pass ,datetime;                                  2e3a3a2
         POINTER u;                                               2e3a3a3
      % lock required users %                                      2e3a3b
         ON ANY SIGNAL                                            2e3a3b1
            BEGIN                                                 2e3a3b1a
            UNLOCK usrlst;                                        2e3a3b1b
            #UNLCK-USRS# (rqstr ! REGISTRARS ! self,WRIT !
             usr ! usrset) @ self;                                2e3a3b1c
            END;                                                  2e3a3b1d
         #LCK-USRS# (rqstr ! REGISTRARS ! self,WRIT ,RETR) @
          self;                                                   2e3a3b2
      % verify requestor's identity %                             2e3a3c
         VER-IDNTTY-USRS (rqstr);                                 2e3a3c1
      % verify write access to registrar %                        2e3a3d
         IF VER-MEM-USRS (REGISTRARS ,rqstr)                      2e3a3d1
            THEN slave _ TRUE                                     2e3a3d1a
         ELSE                                                     2e3a3d2
            BEGIN                                                 2e3a3d2a
            VER-MEM-USRS (self,WRIT ,rqstr);                      2e3a3d2b
            slave _ FALSE;                                        2e3a3d2c
            END;                                                  2e3a3d2d
      % clean names %                                             2e3a3e
         #PRS-USRS# (usr ! usrnm ,pass);                          2e3a3e1
         #PRS-USRS# (memusrs ! memnms);                           2e3a3e2
```

```
   #PRS-USRS# (readusrs : readnms);                              2e3a3e3
   #PRS-USRS# (writusrs : writnms);                              2e3a3e4
   #PRS-USRS# (appusrs : appnms);                                2e3a3e5
   #PRS-USRS# (delusrs : delnms);                                2e3a3e6
   #PRS-USRS# (ctrlusrs : ctrlnms);                              2e3a3e7
   #PRS-USRS# (rqstr : rqnm);                                    2e3a3e8
   #PRS-USRS# (sphere : sphnm);                                  2e3a3e9
   IF slave THEN #PRS-USRS# (crtusr : crtnm);                   2e3a3e10
% create shorthands %                                            2e3a3f
   usrset _ memnms ! readnms ! writnms ! appnms !
    delnms ! ctrlnms ! sphnms;                                   2e3a3f1
% lock new user %                                               2e3a3g
   #LCK-USRS# (usr ,CRT);                                        2e3a3g1
% verify existence of membership ! access lists !
 sphere %                                                        2e3a3h
   #LCK-USRS# (usrset ,XST);                                     2e3a3h1
% verify sphere %                                               2e3a3i
   VER-MEM-USRS (RECORDERS ,sphere);                             2e3a3i1
% create user locally %                                         2e3a3j
   LOCK usrlst;                                                  2e3a3j1
   usrlst [BUMP numusrs] _ u _ ALLOC userentry;                  2e3a3j2
   u.name _ usrnm;                                               2e3a3j3
   u.password _ pass;                                            2e3a3j4
   u.readdate _ null;                                            2e3a3j5
   u.writedate _ null;                                           2e3a3j6
   u.usedate _ null;                                             2e3a3j7
   u.membership _ memnms;                                        2e3a3j8
   u.affiliations _ null;                                        2e3a3j9
   u.ral _ readnms;                                             2e3a3j10
   u.wal _ writnms;                                             2e3a3j11
   u.aal _ appnms;                                              2e3a3j12
   u.xel _ delnms;                                              2e3a3j13
   u.cal _ ctrlnms;                                             2e3a3j14
   IF slave THEN                                                2e3a3j15
      BEGIN                                                    2e3a3j15a
      u.creator _ crtnm;                                       2e3a3j15b
      u.createdate _ crtdte;                                   2e3a3j15c
      u.home _ rqnm;                                           2e3a3j15d
      END;                                                     2e3a3j15e
   ELSE                                                         2e3a3j16
      BEGIN                                                    2e3a3j16a
      u.creator _ rqnm;                                        2e3a3j16b
      u.createdate _ DATETIME;                                 2e3a3j16c
      u.home _ self;                                           2e3a3j16d
      END;                                                     2e3a3j16e
   date _ u.createdate;                                         2e3a3j17
   UNLOCK usrlst;                                               2e3a3j18
```

```
% include new user among membership's affiliations %   2e3a3k
   #LCK=USRS# (memusrs ,WRIT);                         2e3a3k1
   LOCK usrlst;                                        2e3a3k2
   FOR EVERY n IN memnms DO                            2e3a3k3
      BEGIN                                            2e3a3k3a
      u _ usrlst [FIND (n)];                           2e3a3k3b
      u.affiliations _ u.affiliations ! usrnm;         2e3a3k3c
      END;                                             2e3a3k3d
% create user remotely if not already being done %     2e3a31
   IF NOT slave THEN                                   2e3a311
      BROADCAST (sphere ,#CRT=USR# (usr ,memusrs
       ,readusrs ,writusrs ,appusrs ,delusrs ,ctrlusrs
       ,self ,rqstr ,date));                          2e3a311a
% return %                                             2e3a3m
   EXECUTE SIGNAL;                                     2e3a3m1
   RETURN (YES);                                       2e3a3m2
   END.                                                2e3a3m3
```

```
DEL-USRS (usrs ,*rqstr ! YES/err)                                    2e3b

   Requestor:  another Registrar
                  -- or --
               anyone with delete access to the users
                 and write access to the Registrar                   2e3b1

   Description:                                                      2e3b2

      This primitive deletes the users whose names are
      specified.                                                     2e3b2a

   Pseudo-L10 Implementation:                                        2e3b3

      % locals %                                                     2e3b3a
        LOCAL i ,slave;                                             2e3b3a1
        LOCAL STRING n ,rqnm ,usrsnms;                              2e3b3a2
        POINTER u;                                                  2e3b3a3
      % lock required users %                                       2e3b3b
        ON ANY SIGNAL                                               2e3b3b1
           BEGIN                                                    2e3b3b1a
           UNLOCK usrlst;                                           2e3b3b1b
           #UNLCK-USRS# (rqstr ! REGISTRARS ! self,WRIT !
            usrs) @ self;                                           2e3b3b1c
           END;                                                     2e3b3b1d
        #LCK-USRS# (rqstr ! REGISTRARS ! self,WRIT ,RETR) @
         self;                                                      2e3b3b2
      % verify requestor's identity %                               2e3b3c
        VER-IDNTTY-USRS (rqstr);                                    2e3b3c1
      % verify write access to registrar %                          2e3b3d
        IF VER-MEM-USRS (REGISTRARS ,rqstr) THEN                    2e3b3d1
           slave _ TRUE                                             2e3b3d1a
        ELSE                                                        2e3b3d2
           BEGIN                                                    2e3b3d2a
           VER-MEM-USRS (self,WRIT ,rqstr);                         2e3b3d2b
           slave_ FALSE;                                            2e3b3d2c
           END;                                                     2e3b3d2d
      % clean names %                                               2e3b3e
        #PRS-USRS# (rqstr ! rqnm);                                  2e3b3e1
        #PRS-USRS# (usrs ! usrsnms);                                2e3b3e2
      % lock users %                                                2e3b3f
        #LCK-USRS# (usrs ,DEL);                                     2e3b3f1
      % verify delete access %                                      2e3b3g
        LOCK usrlst;                                                2e3b3g1
        FOR EVERY n IN usrsnms DO                                   2e3b3g2
           BEGIN                                                    2e3b3g2a
           u _ usrlst [FIND (n)];                                   2e3b3g2b
```

```
      IF NOT (rqnm AMONG u,xal) THN SIGNAL (err);          2e3b3g2c
      END;                                                 2e3b3g2d
% delete users locally %                                    2e3b3h
   FOR i _ numusrs DOWN UNTIL 0 DO                         2e3b3h1
      BEGIN                                                2e3b3h1a
      u _ usrlst [i];                                      2e3b3h1b
      IF (u,name AMONG usrsnms) THEN                       2e3b3h1c
         BEGIN                                            2e3b3h1c1
         DEALLOC u;                                       2e3b3h1c2
         usrlst [i] _ usrlst [numusrs];                   2e3b3h1c3
         BUMP DOWN numuusrs;                              2e3b3h1c4
         END;                                             2e3b3h1c5
      END;                                                 2e3b3h1d
   FOR i _ 0 UNTIL numusrs DO                              2e3b3h2
      BEGIN                                                2e3b3h2a
      u _ usrlst [i];                                      2e3b3h2b
      REMOVE usrsnms FROM u,membership;                    2e3b3h2c
      REMOVE usrsnms FROM u,ral;                           2e3b3h2d
      REMOVE usrsnms FROM u,wal;                           2e3b3h2e
      REMOVE usrsnms FROM u,aal;                           2e3b3h2f
      REMOVE usrsnms FROM u,xal;                           2e3b3h2g
      REMOVE usrsnms FROM u,cal;                           2e3b3h2h
      END;                                                 2e3b3h2i
% delete user remotely if not already being done %          2e3b3i
   IF NOT slave THEN                                       2e3b3i1
      BROADCAST (REGISTRARS ,#DEL-USRS# (usrs ,self));
                                                           2e3b3i1a
% return %                                                  2e3b3j
   EXECUTE SIGNAL;                                         2e3b3j1
   RETURN (YES);                                           2e3b3j2
   END.                                                    2e3b3j3
```

```
LOGIN-USR (*usr ,password/(*rqstr ,date) : YES/err)                    2e3c

   Requestor:  anyone whose home is the Registrar
               or the user's home Registrar                            2e3c1

   Description:                                                        2e3c2

      This primitives notifies the Registrar of the
      requesting user's intent to use the System,  The user
      presents his permanent password and a proposed,
      short-term password which he will present as
      identificaton in all succeeding procedure calls during
      this login period,                                              2e3c2a

   Pseudo-L10 Implementation:                                          2e3c3

      % locals %                                                       2e3c3a
         LOCAL STRING nm, rqnm;                                       2e3c3a1
         POINTER u;                                                   2e3c3a2
      % lock required users %                                         2e3c3b
         ON ANY SIGNAL                                               2e3c3b1
            BEGIN                                                    2e3c3b1a
            #UNLCK-USRS# (usr) @ self;                               2e3c3b1b
            IF rqstr THEN #UNLCK-USRS# (rqstr ! REGISTRARS);
                                                                     2e3c3b1c
            END;                                                     2e3c3b1d
         #LCK-USRS# (usr ,CHNG) @ self;                              2e3c3b2
      % verify [requestor's] identity %                              2e3c3c
         IF rqstr THEN                                               2e3c3c1
            BEGIN                                                    2e3c3c1a
            #LCK-USRS# (rqstr ! REGISTRARS ,RETR);                   2e3c3c1b
            #VER-IDNTTY-USRS# (rqstr ,self);                         2e3c3c1c
            #VER-MEM-USRS# (rqstr ,REGISTRARS ,self);                2e3c3c1d
            END;                                                     2e3c3c1e
      % clean names %                                                 2e3c3d
         #PRS-USRS# (usr ! nm ,pss);                                 2e3c3d1
         #PRS-USRS# (rqstr ! rqnm);                                  2e3c3d2
      % [verify password] %                                          2e3c3e
         LOCK usrlst;                                                2e3c3e1
         u _ usrlst [FIND (nm)];                                     2e3c3e2
         UNLOCK usrlst;                                              2e3c3e3
         IF password THEN                                            2e3c3e4
            IF u.password # password THEN SIGNAL (err);              2e3c3e4a
      % check for duplicate password %                               2e3c3f
         IF (pss AMONG u.runtimepasswords) THEN SIGNAL
          (err);                                                     2e3c3f1
      % verify user's home %                                         2e3c3g
```

```
   IF u,home # (IF rqstr THEN rqnm ELSE self) THEN     2e3c3g1
      SIGNAL (err);                                     2e3c3g1a
% log the user in %                                     2e3c3h
   u,runtimepasswords _ u,runtimepasswords ! pss;       2e3c3h1
   u,usedate _ IF rqstr THEN date ELSE DATETIME;        2e3c3h2
% notify other registrars %                            2e3c3i
   IF rqstr THEN BROADCAST (u,sphere ,#LOGIN=USR# (usr
    ,self));                                            2e3c3i1
% return %                                             2e3c3j
   EXECUTE SIGNAL;                                      2e3c3j1
   RETURN (YES);                                        2e3c3j2
   END,                                                 2e3c3j3
```

```
LOGOUT-USR (*usr [,*rqstr] : YES/err)                                  2e3d

   Requestor:  anyone whose home is the Registrar
               or the user's home Registrar                            2e3d1

   Description:                                                        2e3d2

      This primitives notifies the Registrar of the
      requesting user's intent to leave the System,  Once
      this call has been performed, the short-term password
      assiged at login becomes invalid,                               2e3d2a

   Pseudo-L10 Implementation:                                          2e3d3

      % locals %                                                       2e3d3a
        LOCAL STRING nm, rqnm;                                        2e3d3a1
        POINTER u;                                                    2e3d3a2
      % lock required users %                                          2e3d3b
        ON ANY SIGNAL                                                 2e3d3b1
           BEGIN                                                     2e3d3b1a
           #UNLCK-USRS# (usr) @ self;                                2e3d3b1b
           IF rqstr THEN #UNLCK-USRS# (rqstr ! REGISTRARS);
                                                                     2e3d3b1c
           END;                                                      2e3d3b1d
        #LCK-USRS# (usr ,CHNG) @ self;                               2e3d3b2
      % verify [requestor's] identity %                              2e3d3c
        IF rqstr THEN                                                2e3d3c1
           BEGIN                                                     2e3d3c1a
           #LCK-USRS# (rqstr ! REGISTRARS ,RETR);                    2e3d3c1b
           #VER-IDNTTY-USRS# (rqstr ,self);                          2e3d3c1c
           #VER-MEM-USRS# (rqstr ,REGISTRARS ,self);                 2e3d3c1d
           END;                                                      2e3d3c1e
      % verify user's identity %                                      2e3d3d
        #VER-IDNTTY-USRS# (usr);                                     2e3d3d1
      % clean names %                                                 2e3d3e
        #PRS-USRS# (usr ! nm);                                       2e3d3e1
        #PRS-USRS# (rqstr ! rqnm);                                   2e3d3e2
      % verify user's home %                                         2e3d3f
        LOCK usrlst;                                                 2e3d3f1
        u _ usrlst [FIND (nm)];                                      2e3d3f2
        UNLOCK usrlst;                                               2e3d3f3
        IF u.home # (IF rqstr THEN rqnm ELSE self) THEN              2e3d3f4
           SIGNAL (err);                                             2e3d3f4a
      % log the user out %                                           2e3d3g
        REMOVE pss FROM u.runtimepasswords;                          2e3d3g1
      % notify other registrars %                                    2e3d3h
```

```
    IF rqstr THEN BROADCAST (u,sphere ,#LOGOUT=USR#                    2e3d3h1
     (usr ,self));                                                      2e3d3i
% return %                                                             2e3d3i1
    EXECUTE SIGNAL;                                                    2e3d3i2
    RETURN (YES);                                                      2e3d3i3
    END,
```

```
CHNG-MEM-USR (usr ,ADD/DEL/REPL ,memusrs ,*rqstr ! YES/err)     2e3e

   Requestor:  another Registrar
                  -- or --
               anyone with write access to the user
               (for ADD, append access will suffice)
                 and with write access to the Registrar        2e3e1

   Description:                                                2e3e2

      This primitive modifies the membership of the
      specified user,                                          2e3e2a

         Depending upon the operation selected, the
         primitive adds the specified users to the
         membership list, deletes them from it, or replaces
         it with them,                                         2e3e2a1

   Pseudo-L10 Implementation:                                  2e3e3

      % locals %                                               2e3e3a
        LOCAL slave;                                           2e3e3a1
        LOCAL STRING usrnm ,memnms ,rqnm ,n;                   2e3e3a2
        POINTER u;                                             2e3e3a3
      % lock required users %                                  2e3e3b
        ON ANY SIGNAL                                          2e3e3b1
           BEGIN                                               2e3e3b1a
           UNLOCK usrlst;                                      2e3e3b1b
           #UNLCK-USRS# (rqstr ! REGISTRARS ! self,WRIT !
            usr) @ self;                                       2e3e3b1c
           END;                                                2e3e3b1d
        #LCK-USRS# (rqstr ! REGISTRARS ! self,WRIT ,RETR) @
         self;                                                 2e3e3b2
      % verify requestor's identity %                          2e3e3c
        VER-IDNTTY-USRS (rqstr);                               2e3e3c1
      % verify write access to registrar %                     2e3e3d
        IF VER-MEM-USRS (REGISTRARS ,rqstr) THEN               2e3e3d1
           slave _ TRUE                                        2e3e3d1a
        ELSE                                                   2e3e3d2
           BEGIN                                               2e3e3d2a
           VER-MEM-USRS (self,WRIT ,rqstr);                    2e3e3d2b
           slave _ FALSE;                                      2e3e3d2c
           END;                                                2e3e3d2d
      % clean names %                                          2e3e3e
        #PRS-USRS# (usr ! usrnm);                              2e3e3e1
        #PRS-USRS# (memusrs ! memnms);                         2e3e3e2
        #PRS-USRS# (rqstr ! rqnm);                             2e3e3e3
```

```
% lock user %                                                    2e3e3f
   #LCK-USRS# (usr ,CHNG);                                      2e3e3f1
% verify write access %                                         2e3e3g
   u _ usrlst [FIND (usrnm)];                                   2e3e3g1
   IF NOT (rqnm AMONG u,wal) THEN SIGNAL (err);                 2e3e3g2
% verify existence of membership %                              2e3e3h
   FOR EVERY n IN memnms DO FIND (n);                           2e3e3h1
% change membership locally %                                   2e3e3i
   CHNG-LIST (operation ,memnms ,u,mem);                        2e3e3i1
   UNLOCK usrlst;                                               2e3e3i2
% change membership remotely if not already being done
 %                                                              2e3e3j
   IF NOT slave THEN                                            2e3e3j1
      BROADCAST (REGISTRARS ,#CHNG-MEM-USR# (usr
       ,operation ,memusrs ,self));                             2e3e3j1a
% return %                                                      2e3e3k
   EXECUTE SIGNAL;                                              2e3e3k1
   RETURN (YES);                                                2e3e3k2
   END,                                                         2e3e3k3
```

```
RETR-MEM-USR (usr ,*rqstr : (YES ,memusrs) / err)                    2e3f

   Requestor:  anyone with read access to the user
                 and to the Registrar                                2e3f1

   Description:                                                      2e3f2

      This primitive retrieves the membership of the
      specified user.                                                2e3f2a

   Pseudo-L10 Implementation:                                        2e3f3

      % locals %                                                     2e3f3a
         LOCAL STRING usrnm ,people;                                 2e3f3a1
         POINTER u;                                                  2e3f3a2
      % lock required users %                                        2e3f3b
         ON ANY SIGNAL                                               2e3f3b1
            BEGIN                                                    2e3f3b1a
            UNLOCK usrlst;                                           2e3f3b1b
            #UNLCK-USRS# (rqstr ! self,READ ! usr) @ self;           2e3f3b1c
            END;                                                     2e3f3b1d
         #LCK-USRS# (rqstr ! self,READ ! usr ,RETR) @ self;          2e3f3b2
      % verify requestor's identity %                                2e3f3c
         VER-IDNTTY-USRS (rqstr);                                    2e3f3c1
      % verify read access to registrar %                            2e3f3d
         VER-MEM-USRS (self,READ ,rqstr);                           2e3f3d1
      % clean names %                                                2e3f3e
         #PRS-USRS# (usr : usrnm);                                   2e3f3e1
      % fetch membership %                                           2e3f3f
         u _ usrlst [FIND (usrnm)];                                  2e3f3f1
         people _ u.membership;                                      2e3f3f2
      % return %                                                     2e3f3g
         EXECUTE SIGNAL;                                             2e3f3g1
         RETURN (YES ,people);                                       2e3f3g2
         END.                                                        2e3f3g3
```

```
VER-MEM-USRS (usrs ,testusrs ,*rqstr : YES/err)                    2e3g

  Requestor:  anyone with read access to the user
                and to the registrar                               2e3g1

  Description:                                                     2e3g2

    This primitive verifies that the specified test users
    are members, directly or indirectly, of at least one
    of the specified users.                                        2e3g2a

  Pseudo-L10 Implementation:                                       2e3g3

    % locals %                                                     2e3g3a
      LOCAL STRING people ,testnms;                                2e3g3a1
    % fetch membership %                                           2e3g3b
      RETR-MEM-USR (usr : people) @ self;                          2e3g3b1
    % clean names %                                                2e3g3c
      #PRS-USRS# (testusrs : testnms);                             2e3g3c1
    % check for test users %                                       2e3g3d
      IF NOT (testnms AMONG people) THEN SIGNAL (err);             2e3g3d1
    % return %                                                     2e3g3e
      RETURN (YES);                                                2e3g3e1
      END.                                                         2e3g3e2
```

```
CHNG-ACC-USR (usr ,ADD/DEL/REPL ,READ/WRIT/APP/DEL/CTRL
,accusrs ,*rqstr : YES/err)                                    2e3h
```

Requestor: another Registrar
-- or --
anyone with controlling access to the user
and write access to the registrar 2e3h1

Description: 2e3h2

This primitive modifies the read, write, append, delete, or controlling access list for the specified user, 2e3h2a

Depending upon the operation selected, the primitive adds the specified users to the access list, deletes them from it, or replaces it with them, 2e3h2a1

Pseudo-L10 Implementation: 2e3h3

```
% locals %                                                    2e3h3a
   LOCAL slave ,alp;                                         2e3h3a1
   LOCAL STRING usrnm ,accnms ,rqnm ,n;                      2e3h3a2
% lock required users %                                      2e3h3b
   ON ANY SIGNAL                                             2e3h3b1
      BEGIN                                                 2e3h3b1a
      UNLOCK usrlst;                                        2e3h3b1b
      #UNLCK-USRS# (rqstr ! REGISTRARS ! self,WRIT !
       usr) @ self;                                         2e3h3b1c
      END;                                                  2e3h3b1d
   #LCK-USRS# (rqstr ! REGISTRARS ! self,WRIT ,RETR) @
    self;                                                   2e3h3b2
% verify requestor's identity %                             2e3h3c
   VER-IDNTTY-USRS (rqstr);                                 2e3h3c1
% verify write access to registrar %                        2e3h3d
   IF VER-MEM-USRS (REGISTRARS ,rqstr) THEN                 2e3h3d1
      slave _ TRUE                                          2e3h3d1a
   ELSE                                                     2e3h3d2
      BEGIN                                                 2e3h3d2a
      VER-MEM-USRS (self,WRIT ,rqstr);                      2e3h3d2b
      slave _ FALSE;                                        2e3h3d2c
      END;                                                  2e3h3d2d
% clean names %                                             2e3h3e
   #PRS-USRS# (usr ! usrnm);                                2e3h3e1
   #PRS-USRS# (accusrs ! accnms);                           2e3h3e2
   #PRS-USRS# (rqstr ! rqnm);                               2e3h3e3
```

```
% lock user %                                                    2e3h3f
   #LCK-USRS# (usr ,CHNG);                                      2e3h3f1
% verify existence and controlling access %                      2e3h3g
   u _ FIND (usrnm);                                            2e3h3g1
   IF NOT (rqnm AMONG u,cal) THEN SIGNAL (err);                 2e3h3g2
% verify existence of new access users %                        2e3h3h
   FOR EVERY n IN accnms DO FIND (n);                           2e3h3h1
% change access list locally %                                  2e3h3i
   CASE accesstype OF                                           2e3h3i1
      = READ ; alp _ $ u,ral;                                   2e3h3i1a
      = WRIT ; alp _ $ u,wal;                                   2e3h3i1b
      = APP  ; alp _ $ u,aal;                                   2e3h3i1c
      = DEL  ; alp _ $ u,xal;                                   2e3h3i1d
      = CTRL ; alp _ $ u,cal;                                   2e3h3i1e
      ENDCASE SIGNAL (err);                                     2e3h3i1f
   CHNG-LIST (operation ,accnms ,[alp]);                        2e3h3i2
   UNLOCK usrlst;                                               2e3h3i3
% change access list remotely if not already being
 done %                                                         2e3h3j
   IF NOT slave THEN                                            2e3h3j1
      BROADCAST (REGISTRARS ,#CHNG-ACC-USR# (usr
       ,operation ,accesstype ,accusrs ,self));                 2e3h3j1a
% return %                                                      2e3h3k
   EXECUTE SIGNAL;                                              2e3h3k1
   RETURN (YES);                                                2e3h3k2
   END.                                                         2e3h3k3
```

```
RETR-ACC-USR (usr ,READ/WRIT/APP/DEL/CTRL ,*rqstr : (YES                    2e3i
,accusrs) / err)

   Requestor:  anyone with read access to the user
               and to the Registrar                                          2e3i1

   Description:                                                              2e3i2

      This primitive retrieves the specified access list for
      the specified user,                                                    2e3i2a

   Pseudo-L10 Implementation:                                                2e3i3

      % locals %                                                             2e3i3a
         LOCAL STRING usrnm ,people;                                         2e3i3a1
         POINTER u;                                                          2e3i3a2
      % lock required users %                                                2e3i3b
         ON ANY SIGNAL                                                       2e3i3b1
            BEGIN                                                            2e3i3b1a
            UNLOCK usrlst;                                                   2e3i3b1b
            #UNLCK-USRS# (rqstr ! self,READ ! usr) @ self;                   2e3i3b1c
            END;                                                             2e3i3b1d
         #LCK-USRS# (rqstr ! self,READ ! usr ,RETR) @ self;                  2e3i3b2
      % verify requestor's identity %                                        2e3i3c
         VER-IDNTTY-USRS (rqstr);                                            2e3i3c1
      % verify read access to registrar %                                    2e3i3d
         VER-MEM-USRS (self,READ ,rqstr);                                   2e3i3d1
      % clean names %                                                        2e3i3e
         #PRS-USRS# (usr ! usrnm);                                           2e3i3e1
      % verify user's existence %                                           2e3i3f
         u _ usrlst [FIND (usrnm)];                                          2e3i3f1
      % fetch appropriate access list %                                     2e3i3g
         CASE accesstype OF                                                  2e3i3g1
            =READ: people _ u,ral;                                           2e3i3g1a
            =WRIT: people _ u,wal;                                           2e3i3g1b
            =APP:  people _ u,aal;                                           2e3i3g1c
            =DEL:  people _ u,xal;                                           2e3i3g1d
            =CTRL: people _ u,cal;                                           2e3i3g1e
            ENDCASE SIGNAL (err);                                            2e3i3g1f
      % return %                                                            2e3i3h
         EXECUTE SIGNAL;                                                     2e3i3h1
         RETURN (YES ,people);                                               2e3i3h2
         END,                                                                2e3i3h3
```

```
VER-ACC-USR (usr ,READ/WRIT/APP/DEL/CTRL ,testusrs ,*rqstr :
YES/err)                                                    2e3j

   Requestor:  anyone with read access to the user
                 and to the Registrar                       2e3j1

   Description:                                             2e3j2

      This primitive verifies that the specified users have
      read, write, append, delete, or controlling access to
      the specified user.                                   2e3j2a

   Pseudo-L10 Implementation:                               2e3j3

      % locals %                                            2e3j3a
         LOCAL STRING people ,testnms;                      2e3j3a1
      % fetch access list %                                 2e3j3b
         RETR-ACC-USR (usr ,accesstype : people) @ self;    2e3j3b1
      % clean names %                                       2e3j3c
         #PRS-USRS# (testusrs : testnms);                   2e3j3c1
      % check for test users %                             2e3j3d
         IF NOT (testnms AMONG people) THEN SIGNAL (err);   2e3j3d1
      % return %                                           2e3j3e
         RETURN (YES);                                      2e3j3e1
         END.                                               2e3j3e2
```

```
VER-XST-USRS (usrs ,*rqstr : YES/err)                                  2e3k

   Requestor:  anyone with read access to the Registrar                2e3k1

   Description:                                                        2e3k2

      This primitive verifies the existence of the specified
      users.                                                           2e3k2a

   Pseudo-L10 Implementation:                                          2e3k3

      % lock required users %                                          2e3k3a
        ON ANY SIGNAL #UNLCK-USRS# (rqstr ! self,READ !
         usrs) @ self;                                                 2e3k3a1
        #LCK-USRS# (rqstr ! self,READ ,RETR) @ self;                   2e3k3a2
      % verify requestor's identity %                                  2e3k3b
        VER-IDNTTY-USRS (rqstr);                                       2e3k3b1
      % verify read access to registrar %                              2e3k3c
        VER-MEM-USRS (self,READ ,rqstr);                              2e3k3c1
      % verify users' existence %                                      2e3k3d
        #LCK-USRS# (usrs ,XST);                                        2e3k3d1
      % return %                                                       2e3k3e
        EXECUTE SIGNAL;                                                2e3k3e1
        RETURN (YES);                                                  2e3k3e2
        END.                                                           2e3k3e3
```

```
VER-IDNTTY-USRS (usrs ,*rqstr : YES/err)                          2e31

   Requestor:  anyone with read access to the Registrar           2e311

   Description:                                                   2e312

      This primitive verifies the identity of the specified
      users.                                                      2e312a

   Pseudo-L10 Implementation:                                     2e313

      % lock required users %                                     2e313a
        ON ANY SIGNAL                                            2e313a1
           BEGIN                                                 2e313a1a
           UNLOCK usrlst;                                        2e313a1b
           #UNLCK-USRS# (rqstr ! self,READ ! usrs) @ self;
                                                                 2e313a1c
           END;                                                  2e313a1d
        #LCK-USRS# (rqstr ! self,READ ,RETR) @ self;             2e313a2
      % verify requestor's identity %                             2e313b
        VER-IDNTTY-USRS (rqstr);                                 2e313b1
      % verify read access to registrar %                         2e313c
        VER-MEM-USRS (self,READ ,rqstr);                        2e313c1
      % verify users' identities %                                2e313d
        #LCK-USRS# (usrs ,RETR);                                 2e313d1
        VER-IDNTTY-USRS (usrs);                                  2e313d2
      % return %                                                   2e313e
        EXECUTE SIGNAL;                                          2e313e1
        RETURN (YES);                                            2e313e2
        END.                                                     2e313e3
```

```
PRS-USRS (usrs : (YES ,names ,passes) / err)                    2e3m

   Requestor:  anyone                                           2e3m1

   Description:                                                 2e3m2

      This primitive separates the name and password lists
      from the specified list of users.                         2e3m2a

   Pseudo-L10 Implementation:                                   2e3m3

      % locals %                                                2e3m3a
         LOCAL STRING cumnam ,cumpas ,nxt ,np;                  2e3m3a1
         LOCAL TEXT POINTER tp1 ,tp2;                           2e3m3a2
      % parse user list %                                       2e3m3b
         cumnam _ cumpas _ null;                                2e3m3b1
         FOR EVERY np IN usrs DO                                2e3m3b2
            BEGIN                                               2e3m3b2a
            FIND SF (np) ^tp1 $LD ^tp2 ('/ / ENDCHR);           2e3m3b2b
            nxt _ tp1 tp2;                                      2e3m3b2c
            IF nxt = null THEN SIGNAL (err)                     2e3m3b2d
            ELSE cumnam _ cumnam ! nxt;                         2e3m3b2e
            FIND ^tp1 $LD ^tp2 ENDCHR;                          2e3m3b2f
            nxt _ tp1 tp2;                                      2e3m3b2g
            IF nxt = null THEN nxt _ '?;                        2e3m3b2h
            cumpas _ cumpas ! nxt;                              2e3m3b2i
            END;                                                2e3m3b2j
      % return %                                                2e3m3c
         RETURN (YES ,cumnam ,cumpas);                          2e3m3c1
         END.                                                   2e3m3c2
```

LCK-USRS (usrs ,type ,rqpath,*rqstr [,SLAVE] [,QUE] :
YES/err) 2e3n

Requestor: a Registrar (if SLAVE is specified)
-- or --
a member of SYSTEM 2e3n1

Description: 2e3n2

This primitive either notifies the caller when, or waits him until locks of the specified type can be applied to the specified users. The caller supplies his path through the call stack to distinguish his lock request from those of other instances of himself, and to avoid waiting hopelessly for pending locks imposed by procedures in his own call stack. 2e3n2a

Pseudo-L10 Implementation: 2e3n3

```
% locals %                                                    2e3n3a
  LOCAL STRING masusr, ackusr;                               2e3n3a1
% lock required users %                                      2e3n3b
  ON ANY SIGNAL UNLCK (rqstr ! REGISTRARS ! SYSTEM
   ,path);                                                   2e3n3b1
  LCK (rqstr ! REGISTRARS ! SYSTEM ,RETR ,path);             2e3n3b2
% verify requestor's identity %                             2e3n3c
  VER-IDNTTY-USRS (rqstr);                                   2e3n3c1
% verify locking priviledge %                               2e3n3d
  VER-MEM-USRS (SYSTEM ,rqstr);                              2e3n3d1
% called as slave? %                                        2e3n3e
  IF slave THEN                                              2e3n3e1
    BEGIN                                                    2e3n3e1a
    VER-MEM-USRS (REGISTRARS ,rqstr);                        2e3n3e1b
    masusr _ rqstr;                                          2e3n3e1c
    END;                                                     2e3n3e1d
  ELSE masusr _ null;                                        2e3n3e2
% queue or wait? %                                          2e3n3f
  ackusr _ IF que THEN rqstr ELSE null;                      2e3n3f1
% lock users %                                              2e3n3g
  LCK (usrs ,type ,rqpath ,masusr ,ackusr);                  2e3n3g1
% return %                                                  2e3n3h
  EXECUTE SIGNAL;                                            2e3n3h1
  RETURN (YES);                                              2e3n3h2
  END,                                                       2e3n3h3
```

```
UNLCK-USRS (usrs ,rqpath ,*rqstr [,SLAVE] : YES/err)                    2e3o

   Requestor:  the user that previously locked the users                2e3o1

   Description:                                                         2e3o2

      This primitive unlocks the specified users.                       2e3o2a

   Pseudo-L10 Implementation:                                           2e3o3

      % locals %                                                        2e3o3a
        LOCAL STRING masusr;                                           2e3o3a1
      % lock required users %                                           2e3o3b
        ON ANY SIGNAL UNLCK (rqstr ! REGISTRARS ! SYSTEM
         ,path);                                                       2e3o3b1
        LCK (rqstr ! REGISTRARS ! SYSTEM ,RETR ,path);                 2e3o3b2
      % verify requestor's identity %                                   2e3o3c
        VER-IDNTTY-USRS (rqstr);                                       2e3o3c1
      % verify locking priviledge %                                     2e3o3d
        VER-MEM-USRS (SYSTEM ,rqstr);                                  2e3o3d1
      % called as slave? %                                             2e3o3e
        IF slave THEN                                                  2e3o3e1
           BEGIN                                                       2e3o3e1a
           VER-MEM-USRS (REGISTRARS ,rqstr);                           2e3o3e1b
           masusr _ rqstr;                                             2e3o3e1c
           END;                                                        2e3o3e1d
        ELSE masusr _ null;                                            2e3o3e2
      % unlock users %                                                 2e3o3f
        UNLCK (usrs ,rqpath ,masusr);                                  2e3o3f1
      % return %                                                       2e3o3g
        EXECUTE SIGNAL;                                                2e3o3g1
        RETURN (YES);                                                  2e3o3g2
        END.                                                           2e3o3g3
```

```
LCK (usrs ,type ,rqpath [,masusr] [,ackusr] : YES/err)                2e3p

   *** This is an internal subroutine, ***                            2e3p1

   Description:                                                       2e3p2

      This primitive locks the specified users,                       2e3p2a

   Pseudo-L10 Implementation:                                         2e3p3

      % locals %                                                      2e3p3a
         LOCAL result;                                               2e3p3a1
         POINTER u;                                                  2e3p3a2
      % initiate lock request %                                       2e3p3b
         ENQ-LCK-REQ (usrs ,type ,rqpath ,masusr : u);               2e3p3b1
      % wait for completion %                                        2e3p3c
         ON ANY SIGNAL DEQ-LCK-REQ (u, ABR);                         2e3p3c1
         IF ackusr THEN                                              2e3p3c2
            BEGIN                                                    2e3p3c2a
            CREATE FORK reporter                                     2e3p3c2b
               BEGIN                                                 2e3p3c2b1
               DISABLE SIGNALS;                                      2e3p3c2b2
               result _ WAIT (u);                                    2e3p3c2b3
               IF result # cancelled THEN ACK-LCK-USRS (usrs
                ,rqpath ,result ,self) @ ackusr;                     2e3p3c2b4
               DEQ-LCK-REQ (u ,LV);                                  2e3p3c2b5
               END;                                                  2e3p3c2b6
            START FORK reporter;                                     2e3p3c2c
            END;                                                     2e3p3c2d
         ELSE                                                        2e3p3c3
            BEGIN                                                    2e3p3c3a
            WAIT (u);                                                2e3p3c3b
            DEQ-LCK-REQ (u,LV);                                      2e3p3c3c
            END;                                                     2e3p3c3d
      % return %                                                     2e3p3d
         RETURN (YES);                                               2e3p3d1
         END,                                                        2e3p3d2
```

```
UNLCK (usrs ,rqpath [,masusr] : YES/err)                    2e3q

   *** This is an internal subroutine. ***                  2e3q1

   Description:                                             2e3q2

      This primitive unlocks the specified users.           2e3q2a

   Pseudo-L10 Implementation:                               2e3q3

      % unlock users %                                      2e3q3a
        LCK (usrs ,UNLCK ,rqpath ,masusr);                  2e3q3a1
      % return %                                            2e3q3b
        RETURN (YES);                                       2e3q3b1
        END.                                                2e3q3b2
```

```
CHNG-LIST (ADD/DEL/REPL ,usrs ,list : YES/err)                    2e3r

   *** This is an internal subroutine, ***                        2e3r1

   Description:                                                   2e3r2

      This primitive modifies the specified list of users,        2e3r2a

         Depending upon the operation selected, the
         subroutine adds the specified users to the list,
         deletes them from it, or replaces it with them,          2e3r2a1

   Pseudo-L10 Implementation:                                     2e3r3

      % change user list %                                        2e3r3a
        CASE operation OF                                         2e3r3a1
           =ADD:                                                  2e3r3a1a
              BEGIN                                               2e3r3a1a1
              REMOVE usrs FROM list;                              2e3r3a1a2
              list _ list ! usrs;                                 2e3r3a1a3
              END;                                                2e3r3a1a4
           =DEL:                                                  2e3r3a1b
              BEGIN                                               2e3r3a1b1
              IF NOT (usrs AMONG list) THEN SIGNAL (err);
                                                                  2e3r3a1b2
              REMOVE usrs FROM list;                              2e3r3a1b3
              END;                                                2e3r3a1b4
           =REPL: list _ usrs;                                    2e3r3a1c
           ENDCASE SIGNAL (err);                                  2e3r3a1d
      % return %                                                  2e3r3b
        RETURN (YES);                                             2e3r3b1
        END,                                                      2e3r3b2
```

```
Number Vendor                                                    2f

   FUNCTION:                                                     2f1

      A Number Vendor assigns blocks of ANs to users on request.
      A Number Vendor may only assign ANs that it itself has
      obtained from another Number Vendor, unless it's the
      "master" Number Vendor, in which case it obtained the entire
      AN name space by executive decree.                         2f1a

   DATA STRUCTURES (Pseudo-L10 Globals):                         2f2

      (numentry) RECORD STRING                                   2f2a
         owner, % owner %                                        2f2a1
         ans; % assigned ANs %                                   2f2a2
      DECLARE numnums;                                           2f2b
      DECLARE ARRAY numlst [maxnums];                            2f2c
      DECLARE STRING free;                                       2f2d

   PORTS:                                                        2f3
```

```
GET-NUMS (count ,*rqstr ! (YES ,ans) / err)                          2f3a

   Requestor:  anyone with write access to the Number Vendor         2f3a1

   Description:                                                      2f3a2

      This primitive returns a list of 'count' ANs assigned
      to the requestor.                                              2f3a2a

   Pseudo-L10 Implementation:                                        2f3a3

      % locals %                                                      2f3a3a
        LOCAL i;                                                     2f3a3a1
        LOCAL STRING rqnm ,extras ,result ,n;                        2f3a3a2
        POINTER u;                                                   2f3a3a3
      % lock required users %                                        2f3a3b
        ON ANY SIGNAL                                                2f3a3b1
           BEGIN                                                     2f3a3b1a
           UNLOCK numlst;                                            2f3a3b1b
           #UNLCK-USRS# (rqstr ! self,WRIT) @ anyreg;                2f3a3b1c
           END;                                                      2f3a3b1d
        #LCK-USRS# (rqstr ! self,WRIT ,RETR) @ anyreg;               2f3a3b2
      % verify requestor's identity %                                2f3a3c
        #VER-IDNTTY-USRS# (rqstr ,self);                             2f3a3c1
      % verify write access %                                        2f3a3d
        #VER-MEM-USRSS# (self,WRIT ,rqstr ,self);                    2f3a3d1
      % clean names %                                                2f3a3e
        #PRS-USRS# (rqstr ! rqnm);                                   2f3a3e1
      % locate user's entry %                                        2f3a3f
        LOCK numlst;                                                 2f3a3f1
        IF NOT (u _ numlst [FIND (rqnm)]) THEN                       2f3a3f2
           BEGIN                                                     2f3a3f2a
           numlst [BUMP numnums] _ u _ ALLOC;                        2f3a3f2b
           u,owner _ rqnm;                                           2f3a3f2c
           u,ans _ null;                                             2f3a3f2d
           END;                                                      2f3a3f2e
        UNLOCK numlst;                                               2f3a3f3
      % add to free pool if necessary %                              2f3a3g
        IF (SIZE (free) >= count) THEN extras _ null                 2f3a3g1
        ELSE #GET-NUMS# (maximum + count - SIZE (free)
         ,self ! extras) @ supven;                                   2f3a3g2
        LOCK numlst;                                                 2f3a3g3
        free _ free ! extras;                                        2f3a3g4
      % allocate 'count' ANs from free pool %                        2f3a3h
        i _ 0;                                                       2f3a3h1
        result _ null;                                               2f3a3h2
        FOR EVERY n IN free DO                                       2f3a3h3
```

```
      IF (BUMP i > count) THEN EXIT LOOP                    2f3a3h3a
      ELSE result _ result ! n;                             2f3a3h3b
    REMOVE result FROM free;                                 2f3a3h4
% assign allocated ANs to user %                             2f3a3i
    u,ans _ u,ans ! result;                                  2f3a3i1
% assure supply above safe minimum %                         2f3a3j
    IF SIZE (free) < minimum THEN                            2f3a3j1
      BEGIN                                                  2f3a3j1a
      #GET-NUMS# (maximum - SIZE (free) ,self !
       extras) @ supven;                                     2f3a3j1b
      free _ free ! extras;                                  2f3a3j1c
      END;                                                   2f3a3j1d
% return %                                                    2f3a3k
    EXECUTE SIGNAL;                                          2f3a3k1
    RETURN (YES ,result);                                    2f3a3k2
    END.                                                     2f3a3k3
```

```
RTN-NUMS (ans ,*rqstr : YES/err)                                    2f3b

   Requestor:  the owner of the ANs                                 2f3b1

   Description:                                                     2f3b2

      This primitive returns unused a list of ANs.                  2f3b2a

   Pseudo-L10 Implementation:                                       2f3b3

      % locals %                                                    2f3b3a
         LOCAL i;                                                   2f3b3a1
         LOCAL STRING rqnm ,extras ,n;                              2f3b3a2
         POINTER u;                                                 2f3b3a3
      % lock required users %                                       2f3b3b
         ON ANY SIGNAL                                              2f3b3b1
            BEGIN                                                   2f3b3b1a
            UNLOCK numlst;                                          2f3b3b1b
            #UNLCK-USRS# (rqstr) @ anyreg;                          2f3b3b1c
            END;                                                    2f3b3b1d
         #LCK-USRS# (rqstr ,RETR) @ anyreg;                         2f3b3b2
      % verify requstor's identity %                                2f3b3c
         #VER-IDNTTY-USRS# (rqstr ,self);                           2f3b3c1
      % clean names %                                               2f3b3d
         #PRS-USRS# (rqstr : rqnm);                                 2f3b3d1
      % unassign ANs %                                             2f3b3e
         LOCK numlst;                                               2f3b3e1
         u _ numlst [FIND (rqnm)];                                  2f3b3e2
         IF NOT (ans AMONG u,ans) THEN SIGNAL (err);                2f3b3e3
         REMOVE ans FROM u,ans;                                     2f3b3e4
         free _ free ! ans;                                         2f3b3e5
      % delete user if possible %                                   2f3b3f
         IF u,ans = null THEN                                       2f3b3f1
            FOR i _ 0 UNTIL numnums DO                              2f3b3f1a
               IF numlst [i] = u THEN                               2f3b3f1a1
                  BEGIN                                             2f3b3f1a1a
                  DEALLOC u;                                        2f3b3f1a1b
                  numlst [i] _ numlst [numnums];                    2f3b3f1a1c
                  BUMP DOWN numnums;                                2f3b3f1a1d
                  EXIT LOOP;                                        2f3b3f1a1e
                  END;                                              2f3b3f1a1f
      % assure supply below safe maximum %                          2f3b3g
         IF size (free) > maximum THEN                              2f3b3g1
            BEGIN                                                   2f3b3g1a
            i _ 0;                                                  2f3b3g1b
            extras _ null;                                          2f3b3g1c
            FOR EVERY n IN free DO                                  2f3b3g1d
```

```
            IF (BUMP i > (SIZE (free) - maximum)) THEN
             EXIT LOOP                                          2f3b3g1d1
            ELSE extras _ extras ! n;                           2f3b3g1d2
          REMOVE extras FROM free;                               2f3b3g1e
          UNLOCK numlst;                                         2f3b3g1f
          IF NOT (#RTN-NUMS# (extras , self) @ supven)
           THEN                                                 2f3b3g1g
            BEGIN                                               2f3b3g1g1
            free _ free ! extras;                               2f3b3g1g2
            SIGNAL (err);                                       2f3b3g1g3
            END;                                                2f3b3g1g4
% return %                                                         2f3b3h
    EXECUTE SIGNAL;                                             2f3b3h1
    RETURN (YES);                                               2f3b3h2
    END,                                                        2f3b3h3
```

```
USE-NUMS (ans ,*rqstr :YES/err)                                    2f3c

   Requestor:  a Recorder                                          2f3c1

   Description:                                                    2f3c2

      This primitive irrevocably marks the specified ANs as
      used.                                                        2f3c2a

   Pseudo-L10 Implementation:                                      2f3c3

      % locals %                                                   2f3c3a
         LOCAL i;                                                  2f3c3a1
         LOCAL STRING rqnm ,n;                                     2f3c3a2
         POINTER u;                                                2f3c3a3
      % lock required users %                                      2f3c3b
         ON ANY SIGNAL                                             2f3c3b1
            BEGIN                                                  2f3c3b1a
            UNLOCK numlst;                                         2f3c3b1b
            #UNLCK-USRS# (rqstr ! RECORDERS) @ anyreg;             2f3c3b1c
            END;                                                   2f3c3b1d
         #LCK-USRS# (rqstr ! RECORDERS ,RETR) @ anyreg;            2f3c3b2
      % verify requestor's identity %                              2f3c3c
         #VER-IDNTTY-USRS# (rqstr ,self);                          2f3c3c1
      % verify Recorder %                                          2f3c3d
         #VER-MEM-USRS# (RECORDERS ,rqstr ,self);                  2f3c3d1
      % clean names %                                              2f3c3e
         #PRS-USRS# (rqstr : rqnm);                                2f3c3e1
      % delete ANs from records %                                  2f3c3f
         LOCK numlst;                                              2f3c3f1
         u _ numlst [FIND (rqnm)];                                 2f3c3f2
         IF NOT (ans AMONG u,ans) THEN SIGNAL (err);               2f3c3f3
         REMOVE ans FROM u,ans;                                    2f3c3f4
      % delete user if possible %                                  2f3c3g
         IF u,ans = null THEN                                      2f3c3g1
            FOR i _ 0 UNTIL numnums DO                             2f3c3g1a
               IF numlst [i] = u THEN                              2f3c3g1a1
                  BEGIN                                            2f3c3g1a1a
                  DEALLOC U;                                       2f3c3g1a1b
                  numlst [i] _ numlst [numnums];                   2f3c3g1a1c
                  BUMP DOWN numnums;                               2f3c3g1a1d
                  EXIT LOOP;                                       2f3c3g1a1e
                  END;                                             2f3c3g1a1f
      % notify supplier %                                          2f3c3h
         UNLOCK numlst;                                            2f3c3h1
         #USE-NUMS# (ans ,self) @ supven;                          2f3c3h2
      % return %                                                   2f3c3i
```

```
EXECUTE SIGNAL;                                    2f3c311
RETURN (YES);                                      2f3c312
END,                                               2f3c313
```

```
RETR-OWNS-NUMS (ans : (YES ,usrs) / err)                              2f3d

   Requestor:  anyone with read access to the Number Vendor           2f3d1

   Description:                                                       2f3d2

      This primitive returns the users to whom the specified
      ANs are assigned.                                               2f3d2a

   Pseudo-L10 Implementation:                                         2f3d3

      % locals %                                                      2f3d3a
         LOCAL i;                                                    2f3d3a1
         LOCAL STRING rqnm ,result ,n;                               2f3d3a2
         POINTER u;                                                  2f3d3a3
      % lock required users %                                        2f3d3b
         ON ANY SIGNAL                                               2f3d3b1
            BEGIN                                                    2f3d3b1a
            UNLOCK numlst;                                           2f3d3b1b
            #UNLCK-USRS# (rqstr ! self,READ) @ anyreg;               2f3d3b1c
            END;                                                     2f3d3b1d
         #LCK-USRS# (rqstr ! self,READ ,RETR) @ anyreg;              2f3d3b2
      % verify requestor's identity %                                2f3d3c
         #VER-IDNTTY-USRS# (rqstr ,self);                            2f3d3c1
      % verify read access %                                         2f3d3d
         #VER-MEM-USRS# (self,READ ,rqstr ,self);                    2f3d3d1
      % clean names %                                                2f3d3e
         #PRS-USRS# (rqstr : rqnm);                                  2f3d3e1
      % lookup ANs %                                                 2f3d3f
         result _ null;                                              2f3d3f1
         LOCK numlst;                                                2f3d3f2
         FOR EVERY n IN ans DO                                       2f3d3f3
            BEGIN                                                    2f3d3f3a
            FOR i _ 0 UNTIL numnums DO                               2f3d3f3b
               BEGIN                                                 2f3d3f3b1
               u _ numlst [i];                                       2f3d3f3b2
               IF (n AMONG u,ans) THEN                               2f3d3f3b3
                  BEGIN                                              2f3d3f3b3a
                  result _ result ! u,owner;                         2f3d3f3b3b
                  REPEAT LOOP 2;                                     2f3d3f3b3c
                  END;                                               2f3d3f3b3d
               END;                                                  2f3d3f3b4
            result _ result ! '?;                                    2f3d3f3c
         END;                                                        2f3d3f4
      % return %                                                     2f3d3g
         EXECUTE SIGNAL;                                             2f3d3g1
         RETURN (YES ,result);                                       2f3d3g2
```

JEW 28-MAY-74 16:43 23143
JOURNAL SUBSCRIPTION SERVICE / JEW -- 28 MAY 74
Modules and their Procedure Calls
Number Vendor

END.

2f3d3g3

```
VER-OWN-NUMS (ans ,usr ,*rqstr : YES/err)                          2f3e

   Requestor:  anyone with read access to the Number Vendor        2f3e1

   Description:                                                    2f3e2

      This primitive verifies that the specified ANs are
      assigned to the specified user.                              2f3e2a

   Pseudo-L10 Implementation:                                      2f3e3

      % locals %                                                    2f3e3a
         LOCAL STRING rqnm;                                        2f3e3a1
         POINTER u;                                                2f3e3a2
      % lock required users %                                       2f3e3b
         ON ANY SIGNAL                                             2f3e3b1
            BEGIN                                                  2f3e3b1a
            UNLOCK numlst;                                         2f3e3b1b
            #UNLCK-USRS# (rqstr ! self,READ) @ anyreg;             2f3e3b1c
            END;                                                   2f3e3b1d
         #LCK-USRS# (rqstr ! self,READ ,RETR) @ anyreg;            2f3e3b2
      % verify requestor's identity %                              2f3e3c
         #VER-IDNTTY-USRS# (rqstr ,self);                          2f3e3c1
      % verify read access %                                       2f3e3d
         #VER-MEM-USRS# (self,READ ,rqstr ,self);                  2f3e3d1
      % clean names %                                              2f3e3e
         #PRS-USRS# (rqstr : rqnm);                                2f3e3e1
      % verify ownership %                                         2f3e3f
         LOCK numlst;                                              2f3e3f1
         u _ numlst [FIND (usrnm)];                                2f3e3f2
         IF NOT (ans AMONG u,ans) THEN SIGNAL (err);               2f3e3f3
      % return %                                                   2f3e3g
         EXECUTE SIGNAL;                                           2f3e3g1
         RETURN (YES);                                             2f3e3g2
         END.                                                      2f3e3g3
```

```
CHNG-OWN-NUMS (ans ,usr ,*rqstr ! YES/err)                          2f3f

   Requestor:  the owner of the ANs                                 2f3f1

   Description:                                                     2f3f2

      This primitive reassigns the specified ANs to the
      specified user.                                               2f3f2a

   Pseudo-L10 Implementation:                                       2f3f3

      % locals %                                                    2f3f3a
        LOCAL i;                                                   2f3f3a1
        LOCAL STRING rqnm ,n ,usrnm;                               2f3f3a2
        POINTER u;                                                 2f3f3a3
      % lock required users %                                       2f3f3b
        ON ANY SIGNAL                                              2f3f3b1
           BEGIN                                                   2f3f3b1a
           UNLOCK numlst;                                          2f3f3b1b
           #UNLCK-USRS# (rqstr ! usr) @ anyreg;                    2f3f3b1c
           END;                                                    2f3f3b1d
        #LCK-USRS# (rqstr ! usr ,RETR) @ anyreg;                   2f3f3b2
      % verify requestor's identity %                              2f3f3c
        #VER-IDNTTY-USRS# (rqstr ,self);                           2f3f3c1
      % clean names %                                              2f3f3d
        #PRS-USRS# (rqstr ! rqnm);                                 2f3f3d1
        #PRS-USRS# (usr ! usrnm);                                  2f3f3d2
      % verify ownership %                                         2f3f3e
        LOCK numlst;                                               2f3f3e1
        u _ numlst [FIND (rqnm)];                                  2f3f3e2
        IF NOT (ans AMONG u,ans) THEN SIGNAL (err);                2f3f3e3
        REMOVE ans FROM u,ans;                                     2f3f3e4
      % delete user if possible %                                  2f3f3f
        IF u,ans = null THEN                                       2f3f3f1
           FOR i _ 0 UNTIL numnums DO                              2f3f3f1a
              IF numlst [i] = u THEN                               2f3f3f1a1
                 BEGIN                                             2f3f3f1a1a
                 DEALLOC u;                                        2f3f3f1a1b
                 numlst [i] _ numlst [numnums];                    2f3f3f1a1c
                 BUMP DOWN numnums;                                2f3f3f1a1d
                 EXIT LOOP;                                        2f3f3f1a1e
                 END;                                              2f3f3f1a1f
      % locate new owner's entry %                                 2f3f3g
        IF NOT (u _ numlst [FIND (usrnm)]) THEN                    2f3f3g1
           BEGIN                                                   2f3f3g1a
           numlst [BUMP numnums] _ u _ ALLOC;                      2f3f3g1b
           u,owner _ rqnm;                                         2f3f3g1c
```

```
      u.ans _ null;                                    2f3f3g1d
      END;                                             2f3f3g1e
% assign ANs to new owner %                             2f3f3h
   u.ans _ u.ans ! ans;                                2f3f3h1
% return %                                              2f3f3i
   EXECUTE SIGNAL;                                     2f3f3i1
   RETURN (YES);                                       2f3f3i2
   END.                                                2f3f3i3
```

Retrieval Manager 2g

FUNCTION: 2g1

A Retrieval Manager locates and retrieves the contents of a document on behalf of a user. Although the System assumes the existence of such processes, it says nothing about the algorithm they employ in locating the document. 2g1a

DATA STRUCTURES (Pseudo-L10 Globals): 2g2

None. 2g2a

PORTS: 2g3

```
GET-COPY (an ,*rqstr : (YES ,content) / err)                    2g3a

   Requestor:  anyone with read access to the document          2g3a1

   Description:                                                 2g3a2

      This primitive returns the contents of the document
      whose AN is specified.                                    2g3a2a

   Pseudo-L10 Implementation:                                   2g3a3


      NOTE:  There are any number of possible implemetations
      of this procedure call.  The one presented here first
      verifies that the requesting user has access to the
      document.  If so, the Retrieval Manager checks for a
      local copy of the document.  If none is found, he
      creates one with a lifetime of 30 days.
                                                                2g3a3a

      % locals %                                                2g3a3b
        LOCAL STRING result;                                    2g3a3b1
      % verify requestor's identity %                           2g3a3c
        #VER-IDNTTY-USRS# (rqstr ,self) @ anyreg;               2g3a3c1
      % verify read access %                                    2g3a3d
        #VER-MEM-USRS# (an,READ ,rqstr ,self);                  2g3a3d1
      % check for local image %                                 2g3a3e
        IF NOT (#RETR-IMG# (an ,rqstr : result) @ locrec)
         THEN                                                   2g3a3e1
           BEGIN                                                2g3a3e1a
           #CRT-IMG# (an ,null ,null ,rqstr ,30days);           2g3a3e1b
           #RETR-IMG# (an ,rqstr : result);                     2g3a3e1c
           END;                                                 2g3a3e1d
      % return %                                                2g3a3f
        RETURN (YES ,result);                                   2g3a3f1
        END.                                                    2g3a3f2
```

Delivery Manager 2h

FUNCTION: 2h1

A Delivery Manager accepts delivery of recorded documents and notifies their addressees' of their receipt. 2h1a

DATA STRUCTURES (Pseudo-L10 Globals): 2h2

None. 2h2a

PORTS: 2h3

DELIVER (an, usrs ,*rqstr : YES/err) 2h3a

Requestor: anyone with distribute access to the document 2h3a1

Description: 2h3a2

This primitive causes notice of the document whose AN is specified to be brought to the attention of the specified users, The caller provides the name of a Recorder from whom a copy of the document can be obtained, 2h3a2a

Pseudo-L10 Implementation: 2h3a3

Many possible implementations, 2h3a3a

```
SUMMARY of PROCEDURE CALLS                                                    3
   Storage Manager                                                            3a
      CRT-FILE (pathname ,contents ,readusrs ,writusrs ,delusrs
       ,ctrlusrs ,*rqstr : YES/err)                                           3a1
      RETR-FILE (pathname ,*rqstr : (YES ,contents) / err)                    3a2
      DEL-FILE (pathname ,*rqstr : YES/err)                                   3a3
   Recorder                                                                   3b
      CRT-DOC (an ,numven ,content ,readusrs ,distusrs ,delusrs
       ,ctrlusrs ,*rqstr [,date/interval] : YES/err)                          3b1
      CRT-IMG (an ,delusrs ,ctrlusrs ,*rqstr [,date/interval] :
       YES/err)                                                               3b2
      RETR-IMG (an ,*rqstrs : (YES ,content) / err)                           3b3
      DEL-DOC (an ,*rqstr : YES/err)                                          3b4
      DEL-IMG (an ,*rqstr : YES/err)                                          3b5
   Publisher                                                                  3c
      CRT-ART (an ,citation ,delusrs ,ctrlusrs ,*rqstr : YES/err)             3c1
      DEL-ART (an ,*rqstr : YES/err)                                          3c2
   Cataloger                                                                  3d
      ADD-DOC-CAT (an ,citation ,cntrbtr ,*rqstr : YES/err)                   3d1
      RETR-DOC-CAT (an ,*rqstr : (YES ,citation) / err)                       3d2
      SRCH-CAT (key ,*rqstr : (YES ,ans) / err)                               3d3
      RETR-CAT ,*rqstr : (YES ,catalog) / err)                                3d4
   Registrar                                                                  3e
      CRT-USR (usr ,memusrs ,readusrs ,writusrs ,appusrs ,delusrs
       ,ctrlusrs ,sphere ,*rqstr [,crtusr ,crtdate] : YES/err)                3e1
      DEL-USRS (usrs ,*rqstr : YES/err)                                       3e2
      LOGIN-USR (*usr ,password/(*rqstr ,date) : YES/err)                     3e3
      LOGOUT-USR (*usr [,*rqstr] : YES/err)                                   3e4
      CHNG-MEM-USR (usr ,ADD/DEL/REPL ,memusrs ,*rqstr : YES/err)             3e5
      RETR-MEM-USR (usr ,*rqstr : (YES ,memusrs) / err)                       3e6
      VER-MEM-USRS (usrs ,testusrs ,*rqstr : YES/err)                         3e7
      CHNG-ACC-USR (usr ,ADD/DEL/REPL ,READ/WRIT/APP/DEL/CTRL
       ,accusrs ,*rqstr : YES/err)                                            3e8
      RETR-ACC-USR (usr ,READ/WRIT/APP/DEL/CTRL ,*rqstr : (YES
       ,accusrs) / err)                                                       3e9
      VER-ACC-USR (usr ,READ/WRIT/APP/DEL/CTRL ,testusrs ,*rqstr :
       YES/err)                                                              3e10
      VER-XST-USRS (usrs ,*rqstr : YES/err)                                  3e11
      VER-IDNTTY-USRS (usrs ,*rqstr : YES/err)                               3e12
      PRS-USRS (usrs : (YES ,names ,passes) / err)                           3e13
      LCK-USRS (usrs ,type ,rqpath,*rqstr [,SLAVE] [,QUE] : YES/err)         3e14
      UNLCK-USRS (usrs ,rqpath ,*rqstr [,SLAVE] : YES/err)                   3e15
      LCK (usrs ,type ,rqpath [,masusr] [,ackusr] : YES/err)                 3e16
      UNLCK (usrs ,rqpath [,masusr] : YES/err)                               3e17
      CHNG-LIST (ADD/DEL/REPL ,usrs ,list : YES/err)                         3e18
   Number Vendor                                                              3f
      GET-NUMS (count ,*rqstr : (YES ,ans) / err)                             3f1
```

```
   RTN-NUMS (ans ,*rqstr : YES/err)                                  3f2
   USE-NUMS (ans ,*rqstr :YES/err)                                   3f3
   RETR-OWNS-NUMS (ans : (YES ,usrs) / err)                          3f4
   VER-OWN-NUMS (ans ,usr ,*rqstr : YES/err)                         3f5
   CHNG-OWN-NUMS (ans ,usr ,*rqstr : YES/err)                        3f6
Retrieval Manager                                                     3g
   GET-COPY (an ,*rqstr : (YES ,content) / err)                      3g1
Delivery Manager                                                      3h
   DELIVER (an, usrs ,*rqstr : YES/err)                              3h1
```

SYSTEM IDENTS of INTEREST 4

Created at System Generation 4a

SYSTEM-MANAGER 4a1

Membership: the System Manager 4a1a
Read access: ALL 4a1b
Write access: SELF 4a1c
Append access: SELF 4a1d
Delete access: null 4a1e
Controlling access: SELF 4a1f

RECORDERS 4a2

Membership: all Recorders 4a2a
Read access: ALL 4a2b
Write access: SYSTEM-MANAGER 4a2c
Append access: SYSTEM-MANAGER 4a2d
Delete access: SYSTEM-MANAGER 4a2e
Controlling access: SYSTEM-MANAGER 4a2f

PUBLISHERS 4a3

Membership: all Publishers 4a3a
Read access: ALL 4a3b
Write access: SYSTEM-MANAGER 4a3c
Append access: SYSTEM-MANAGER 4a3d
Delete access: SYSTEM-MANAGER 4a3e
Controlling access: SYSTEM-MANAGER 4a3f

REGISTRARS 4a4

Membership: all Registrars 4a4a
Read access: ALL 4a4b
Write access: SYSTEM-MANAGER 4a4c
Append access: SYSTEM-MANAGER 4a4d
Delete access: SYSTEM-MANAGER 4a4e
Controlling access: SYSTEM-MANAGER 4a4f

CATALOGERS 4a5

Membership: all Catalogers 4a5a
Read access: ALL 4a5b
Write access: SYSTEM-MANAGER 4a5c
Append access: SYSTEM-MANAGER 4a5d
Delete access: SYSTEM-MANAGER 4a5e

Controlling access: SYSTEM-MANAGER 4a5f

NUMBER-VENDORS 4a6

Membership: all Number Vendors 4a6a
Read access: ALL 4a6b
Write access: SYSTEM-MANAGER 4a6c
Append access: SYSTEM-MANAGER 4a6d
Delete access: SYSTEM-MANAGER 4a6e
Controlling access: SYSTEM-MANAGER 4a6f

Created at creation of a Storage Manager 4b

<storman>,MANAGER 4b1

Membership: the Manager of the Storage Manager 4b1a
Read access: ALL 4b1b
Write access: SELF 4b1c
Append access: SELF 4b1d
Delete access: SELF 4b1e
Controlling access: SELF 4b1f

<storman>,WRIT 4b2

Membership: users with write access to the Storage Manager 4b2a
Read access: ALL 4b2b
Write access: <storman>,CTRL 4b2c
Append access: <storman>,CTRL 4b2d
Delete access: <storman>,MANAGER 4b2e
Controlling access: null 4b2f

<storman>,CTRL 4b3

Membership: users with controlling access to the Storage Manager 4b3a
Read access: ALL 4b3b
Write access: SELF 4b3c
Append access: SELF 4b3d
Delete access: <storman>,MANAGER 4b3e
Controlling access: null 4b3f

Created at creation of a Recorder 4c

<recorder>,MANAGER 4c1

Membership: the Manager of the Recorder 4c1a
Read access: ALL 4c1b
Write access: SELF 4c1c
Append access: SELF 4c1d
Delete access: SELF 4c1e
Controlling access: SELF 4c1f

<recorder>,WRIT 4c2

Membership: users with write access to the Recorder 4c2a
Read access: ALL 4c2b
Write access: <recorder>,CTRL 4c2c
Append access: <recorder>,CTRL 4c2d
Delete access: <recorder>,MANAGER 4c2e
Controlling access: null 4c2f

<recorder>,READ 4c3

Membership: users with read access to the Recorder 4c3a
Read access: ALL 4c3b
Write access: <recorder>,CTRL 4c3c
Append access: <recorder>,CTRL 4c3d
Delete access: <recorder>,MANAGER 4c3e
Controlling access: null 4c3f

<recorder>,CTRL 4c4

Membership: users with controlling access to the Recorder 4c4a
Read access: ALL 4c4b
Write access: SELF 4c4c
Append access: SELF 4c4d
Delete access: <recorder>,MANAGER 4c4e
Controlling access: null 4c4f

Created at creation of a Number Vendor 4d

<numven>,MANAGER 4d1

Membership: the Manager of the Number Vendor 4d1a
Read access: ALL 4d1b
Write access: SELF 4d1c
Append access: SELF 4d1d
Delete access: SELF 4d1e
Controlling access: SELF 4d1f

<numven>,WRIT 4d2

Membership: users with write access to the Number Vendor 4d2a
Read access: ALL 4d2b
Write access: <numven>,CTRL 4d2c
Append access: <numven>,CTRL 4d2d
Delete access: <numven>,MANAGER 4d2e
Controlling access: null 4d2f

<numven>,READ 4d3

Membership: users with read access to the Number Vendor 4d3a
Read access: ALL 4d3b
Write access: <numven>,CTRL 4d3c
Append access: <numven>,CTRL 4d3d
Delete access: <numven>,MANAGER 4d3e
Controlling access: null 4d3f

<numven>,CTRL 4d4

Membership: users with controlling access to the Number Vendor 4d4a
Read access: ALL 4d4b
Write access: SELF 4d4c
Append access: SELF 4d4d
Delete access: <numven>,MANAGER 4d4e
Controlling access: null 4d4f

Created at creation of a Publisher 4e

<publisher>,MANAGER 4e1

Membership: the Manager of the Publisher 4e1a
Read access: ALL 4e1b
Write access: SELF 4e1c
Append access: SELF 4e1d
Delete access: SELF 4e1e
Controlling access: SELF 4e1f

<publisher>,WRIT 4e2

Membership: users with write access to the Publisher 4e2a
Read access: ALL 4e2b
Write access: <publisher>,CTRL 4e2c
Append access: <publisher>,CTRL 4e2d
Delete access: <publisher>,MANAGER 4e2e
Controlling access: null 4e2f

<publisher>,READ 4e3

Membership: users with read access to the Publisher 4e3a
Read access: ALL 4e3b
Write access: <publisher>,CTRL 4e3c
Append access: <publisher>,CTRL 4e3d
Delete access: <publisher>,MANAGER 4e3e
Controlling access: null 4e3f

<publisher>,CTRL 4e4

Membership: users with controlling access to the Publisher 4e4a
Read access: ALL 4e4b
Write access: SELF 4e4c
Append access: SELF 4e4d
Delete access: <publisher>,MANAGER 4e4e
Controlling access: null 4e4f

<publisher>,SUBSCRIBERS 4e5

Membership: subscribing catalogers 4e5a
Read access: ALL 4e5b
Write access: SELF 4e5c
Append access: SELF 4e5d
Delete access: <publisher>,MANAGER 4e5e
Controlling access: null 4e5f

```
<publisher>,SUBSCRIBERS                                          4e6

   Membership:  subscribing catalogers                           4e6a
   Read access: ALL                                              4e6b
   Write access: SELF                                            4e6c
   Append access: SELF                                           4e6d
   Delete access: <publisher>,MANAGER                            4e6e
   Controlling access: null                                      4e6f

<publisher>,READERS                                              4e7

   Membership:  subscribers' readers                             4e7a
   Read access: ALL                                              4e7b
   Write access: SELF                                            4e7c
   Append access: SELF                                           4e7d
   Delete access: <publisher>,MANAGER                            4e7e
   Controlling access: null                                      4e7f
```

Created by CRT-FILE -- one set for each file 4f

<storman>,<pathname>,READ 4f1

Membership: users with read access to the file 4f1a
Read access: ALL 4f1b
Write access: <storman>,<pathname>,CTRL 4f1c
Append access: <storman>,<pathname>,CTRL 4f1d
Delete access: <storman> 4f1e
Controlling access: null 4f1f

<storman>,<pathname>,WRIT 4f2

Membership: users with write access to the file 4f2a
Read access: ALL 4f2b
Write access: <storman>,<pathname>,CTRL 4f2c
Append access: <storman>,<pathname>,CTRL 4f2d
Delete access: <storman> 4f2e
Controlling access: null 4f2f

<storman>,<pathname>,DEL 4f3

Membership: users with delete access to the file 4f3a
Read access: ALL 4f3b
Write access: <storman>,<pathname>,CTRL 4f3c
Append access: <storman>,<pathname>,CTRL 4f3d
Delete access: <storman> 4f3e
Controlling access: null 4f3f

<storman>,<pathname>,CTRL 4f4

Membership: users with controlling access to the file 4f4a
Read access: ALL 4f4b
Write access: SELF 4f4c
Append access: SELF 4f4d
Delete access: <storman> 4f4e
Controlling access: null 4f4f

Created by CRT-DOC -- one set for each document 4g

<an>,READ 4g1

Membership: users with read access to the document 4g1a
Read access: ALL 4g1b
Write access: <an>,CTRL 4g1c
Append access: <an>,CTRL ! <an>,DIST 4g1d
Delete access: <an>,HOME 4g1e
Controlling access: null 4g1f

<an>,DIST 4g2

Membership: users with distribute access to the document 4g2a
Read access: ALL 4g2b
Write access: <an>,CTRL 4g2c
Append access: <an>,CTRL 4g2d
Delete access: <an>,HOME 4g2e
Controlling access: null 4g2f

<an>,DEL 4g3

Membership: users with delete access to the document 4g3a
Read access: ALL 4g3b
Write access: <an>,CTRL 4g3c
Append access: <an>,CTRL 4g3d
Delete access: <an>,HOME 4g3e
Controlling access: null 4g3f

<an>,CTRL 4g4

Membership: users with controlling access to the document 4g4a
Read access: ALL 4g4b
Write access: SELF 4g4c
Append access: SELF 4g4d
Delete access: <an>,HOME 4g4e
Controlling access: null 4g4f

<an>,HOME 4g5

Membership: the document's home recorder 4g5a
Read access: ALL 4g5b
Write access: SELF 4g5c
Append access: null 4g5d
Delete access: SELF 4g5e
Controlling access: null 4g5f

<an>,RECORDERS 4g6

Membership: <an>,HOME ! recorders at which images exist 4g6a
Read access: ALL 4g6b
Write access: RECORDERS 4g6c
Append access: RECORDERS 4g6d
Delete access: <an>,HOME 4g6e
Controlling access: null 4g6f

<an>,PUBLISHERS 4g7

Membership: publishers whose journals contain the document 4g7a
Read access: ALL 4g7b
Write access: PUBLISHERS 4g7c
Append access: PUBLISHERS 4g7d
Delete access: <an>,HOME 4g7e
Controlling access: null 4g7f

<an>,READERS 4g8

Membership: readers of the document 4g8a
Read access: ALL 4g8b
Write access: null 4g8c
Append access: RECORDERS 4g8d
Delete access: <an>,HOME 4g8e
Controlling access: null 4g8f

Created by CRT-IMG -- one set for each image 4h

<recorder>,<an>,DEL 4h1

Membership: users with delete access to the image 4h1a
Read access: ALL 4h1b
Write access: <recorder>,<an>,CTRL 4h1c
Append access: <recorder>,<an>,CTRL 4h1d
Delete access: <recorder> 4h1e
Controlling access: null 4h1f

<recorder>,<an>,CTRL 4h2
Membership: users with controlling access to the image 4h2a
Read access: ALL 4h2b
Write access: SELF 4h2c
Append access: SELF 4h2d
Delete access: <recorder> 4h2e
Controlling access: null 4h2f

Created by PUB-DOC -- one set for each document published 4i

<publisher>,<an>,DEL 4i1

Membership: users with delete access to the published document 4i1a
Read access: ALL 4i1b
Write access: <publisher>,<an>,CTRL 4i1c
Append access: <publisher>,<an>,CTRL 4i1d
Delete access: <publisher> 4i1e
Controlling access: null 4i1f

<publisher>,<an>,CTRL 4i2

Membership: users with controlling access to the published document 4i2a
Read access: ALL 4i2b
Write access: SELF 4i2c
Append access: SELF 4i2d
Delete access: <publisher> 4i2e
Controlling access: null 4i2f

Description of a Multi-Host Journal System

Introduction 1

This document is a fairly high-level design discussion of a Multi-Host Journal System (MHJS). It seems unlikely that ARC will be funded to implement such a system, and so the following is offered as a help to would-be implementers of similar systems. 1a

This paper is neither as complete nor as polished as it should be, but in it much of our thinking about what a MHJS should be like has been recorded. 1a1

A companion paper (23143,1) is an attempt to 'implement' each of the program modules discussed in the current paper in a ficticious, L10-like language. 1b

The MHJS is, conceptually, an extension of the present NLS Journal system to embrace an arbitrary number of ARPANET hosts. It's also a new and in many ways different Journal system, in which many of the basic concepts of the present system find a place, but in which also, new concepts appear. 1c

Design Goals 2

MODULARITY 2a

The definition of the MHJS as a multi-host system necessitates that modularity be one of the design goals. 2a1

We desire to specify a system composed of modules, each of which provides some specialized service to the others, or to the end users of the System, and which together comprise a coherent system. 2a2

Each module implements a set of primitives whose syntax and basic function are standardized and advertized, but whose internal workings are left unspecified by the design (within certain broad constraints). The internal functioning of any particular module implementation is dependent upon its host machine, and the particular role which the module is to play within the System as a whole. 2a3

RECONFIGURABILITY 2b

The MHJS is designed to be reconfigurable. Although the design suggests in broad terms the manner in which the System is to be constructed from its component modules, the design does no more than specify a family of MHJSs from which a particular configuration can be selected (in the same way that a computer system manufacturer provides a set of hardware modules (disk

drives, CPUs, etc.) from which the customer configures his particular system). 2b1

The design specifies a small set of module types. An actual MHJS (certainly a truly multi-host one) in all probability contains several (many) instances of each module. 2b2

The MHJS might be reconfigured to accomodate the addition of new hosts to the System, to reduce overhead by moving an instance of a frequently used module closer to a population center, or for any of a variety of other reasons. 2b3

READY ACCESS TO COPIES OF HEAVILY USED DATA BASES 2c

It is, of course, more expensive, in terms both of real and processing time required, to manipulate a data base which resides on another Network host than it is to manipulate a local data base. And, of course, when the distant host is disconnected from the Network (for whatever reason), the data base cannot be accessed at all. A goal of the MHJS design is, therefore, to reduce the frequency with which remote data bases must be dealt with by replicating portions of them in centers of population and to generally minimize the effect of the failure of any of its components upon the System as a whole. 2c1

UNIFORM AND CONSISTENTLY-APPLIED ACCESS CONTROLS 2d

The MHJS must recognize the existence of private information of every type (documents, catalogs, users, etc.) and provide the access controls necessary to protect it. 2d1

Unexplored Areas 3

A number of modules known to be necessary to the MHJS have not been dealt with in this document. And there are a number of areas of concern which have not yet been explored in the design. The reader is referred to (23143,1b) for a brief discussion of some of these areas. 3a

Module Description 4

STORAGE MANAGER 4a

The Need 4a1

Throughout the System a variety of permanent data bases must be maintained (documents, catalogs, user profiles, and so forth), each of which must be housed on one or more physical storage devices. 4a1a

Most of the data bases grow in time, since their size (as it turns out) is in most cases some monotonically increasing function of the number of recorded documents that exist within the System, and since, in many cases, once a document is recorded it is never "unrecorded". 4a1b

Not only must on-line storage be available to the System, but, because of the growth property of the data bases, some form of tertiary storage must also be available. Archiving algorithms must be applied throughout the System to locate heavily used data-base elements on on-line devices and the rest on (hopefully) near-on-line devices. 4a1c

Access to each data base must be appropriately controlled. 4a1d

Module Description 4a2

The System module responsible for management of a particular subset of the physical storage required by the System is called a Storage Manager. 4a2a

Any number of Storage Managers may exist simultaneously within the System, each managing its assigned subset of the System's physical storage. 4a2b

The capacity of a particular Storage Manager may differ greatly from that of another of its kind. One may have very limited capacity, with a single disk as its storage device; another may have almost unlimited storage at its disposal, with a dozen drives or more for frequently accessed files and a laser store for the rest. 4a2b1

But despite the inevitable differences in capacity, type of storage device, and other implementation details, all Storage Managers respond to the same set of primitives and are thus logically interchangeable. This kind of superficial uniformity characterizes all other classes of System modules as well. 4a2b2

Because of its general utility, a storage manager of some sort presumably already exists in most every host in the Network. In the current design, therefore, the primitives assigned for implemetation by the Storage Manager have been chosen to quite closely reflect those of these existing modules, to make them useable with only minor modifications in the MHJS. 4a2c

FILES 4a2d

The Storage Manager trades in commodities called files, each of which is designated by a unique pathname. The Storage Manager specifies the name space from which pathnames may be chosen, and modules which use its services must adhere to that specification. 4a2d1

Pathnames are local handles on a file, and the name spaces of two Storage Managers may therefore overlap unambiguously, both the pathname and the name of the Storage Manager at which the file resides being required to uniquely identify a file within the System. 4a2d1a

A Storage Manager implements primitives which other System modules use to create files, retrieve or replace all or selected components of an existing file, and delete files. 4a2d2

FILE ACCESS CONTROLS 4a2e

The exercise of any file-manipulative primitive by a particular module is subject to file-specific access controls imposed by the Storage Manager. 4a2e1

Assocated with each file are lists of users to whom read, write, delete, and controlling access to that file are respectively granted. 4a2e1a

A user must have read access to a file to invoke the retrieval primitive, delete access to invoke the delete primitive, and so forth. 4a2e1a1

A user with controlling access to a file is permitted to change (by deleting from or adding users to) any of the file's access lists. 4a2e1a2

The primitives by which a module actually effects such changes are not implemented by the Storage Manager, but rather by a module called the Registrar to be described later. 4a2e1b

RESOURCE ACCESS CONTROLS 4a2f

Not only can access to a particular file be regulated, but access to the Storage Manager itself can be regulated. That is, the right to exercise the create-file primitive can be limited to any desired subset of the user population. 4a2f1

By employing this particular type of access control, the (perhaps) limited physical storage managed by a Storage Manager created for exclusive use by a particular System module (call it SMn) can be guaranteed to that module (by including only 'SMn' in the Storage Manager's write access list). 4a2f1a

Another Storage Manager with larger capacity might be created to serve a correspondingly larger clientele, and its access list might grant write access to the entire user population, allowing any user to create and maintain files there. 4a2f1b

RECORDER 4b

The Need 4b1

The prime commodity within the MHJS is the document -- a body of text, uniquely addressable throughout the System by a global handle call a document identifier (DID). 4b1a

For reasons of efficiency and reliability, it's highly desireable to permit an arbitrary number of physical copies of a document, called images, to exist simultaneously, within the System. 4b1b

Each additional image, assuming it's created on a different host, increases the probability of a user's being able to retrieve the document when he wants it. 4b1b1

A retrieval request can be satisfied most quickly, of course, if an image of the requested document happens to exist on the user's own host already. 4b1b2

One strategy, therefore, that the System as a whole might implement is, after recording a document, to create an image of the document at each major population center, anticipating a rash of retrieval requests; and then delete the images a month later, once the period of peak demand has passed. 4b1b2a

Although this particular strategy is probably a good one, the System permits any distribution strategy which, after tinkering with the System's configuration, proves to be effective. 4b1b2b

Access to a document and all its images must be uniformly controlled. The rights to retrieve, delete, and distribute a document must be independently assignable. 4b1c

We also seek to permit the author of a document to conveniently keep tabs on who's read it. 4b1d

Module Description 4b2

The system module which facilitates the orderly creation and manipulation of such networks of document images is called a Recorder. 4b2a

DOCUMENT NETWORKS 4b2b

A document is recorded or an image of it created "at" a particular Recorder. The network of a document's images is therefore a network of Recorders, each of which maintains either the document itself or an image of it. 4b2b1

The creation of document images is a System function designed to promote efficiency and is therefore unhindered by access controls. 4b2b1a

A user can cause the creation of a document image at a Recorder, and yet be unable to retrieve or in any other way affect the image (except to delete it), since access to the document and all of its images is uniformly controlled (by the Recorders), in accordance with the terms specified for the document by its author. 4b2b1a1

The Recorder at which the document itself (as opposed to any of its images) is thought to reside is called the document's home. 4b2b1b

The singling out of one Recorder for this distinction is somewhat artificial, but permits the document's author to specify a set of minimal conditions for the document's storage within the System (i.e., allows him to specify one Recorder with primary responsibility for the document, from which it can always be retrieved). 4b2b1b1

The System maintains a list of the names of all Recorders at which either the document or an image of it exists. By consulting the list, a user can quickly determine where he should go to retrieve the document. 4b2b1c

The Recorder offers primitives for creating and deleting documents, retrieving their contents, for creating and deleting at one Recorder an image of a

document created at another, and for moving a document from one home to another, 4b2b1d

USE OF STORAGE MANAGERS 4b2c

The Recorder stores the text of documents and images at one or more Storage Managers of its choosing. In a sense, it acts as a retailer of file storage by interposing itself between wholesaler (the Storage Manager) and buyer (other System modules), "pre-processing" store, retrieve and delete requests to the wholesaler, 4b2c1

Like any good retailer, the Recorder offers its users conveniences that they wouldn't expect to get from the wholesaler if they dealt directly with him, 4b2c2

One such convenience is that document names can be selected from a System-wide name space, rather than from the Storage Managers' parochial ones, 4b2c2a

Each Recorder implements a mapping between DIDs and the pathnames permitted by the Storage Manager(s) whose services it uses, 4b2c2a1

Another convenience is that the integrity of multiple copies of the document can be maintained, readers kept track of, and so forth, 4b2c2b

DOCUMENT ACCESS CONTROLS 4b2d

The Recorder implements the same kinds of access controls for documents as the Storage Manager does for files -- read, write, delete, and controlling access -- plus one additional type called distribute access, 4b2d1

Only users with distribute access to a document can publish it, mail it to another user, or extend read access to the document to someone who doesn't already have it, 4b2d1a

The primitives by which a module actually effects such changes are not implemented by the Recorder, but rather by a module called the Registrar to be described later, 4b2d2

By appropriately restricting access to the corresponding file at the Storage Manager, the Recorder forces all retrieval requests to be channeled through him, 4b2d3

Even if a user knew the name of the Storage Manager selected by the Recorder to house a particular document, he could not, in an attempt to bypass the Recorder, obtain the text of the document by appealing directly to the Storage Manager; the file can only by retrieved on the Recorder's authority. 4b2d3a

Delete and controlling access to an image of a document can be assigned by the image's creator independently of the document's corresponding access lists, with the one exception that if the document is deleted, all images of it are forcibly deleted, without asking the consent of their creators. 4b2d4

KEEPING TABS ON READERS 4b2e

The System also maintains a list of users who have retrieved a document or any of its images. The Recorder requires a list of users with every retrieve request, and updates the master list accordingly. 4b2e1

PUBLISHER 4c

The Need 4c1

Since users, in general, create documents to be read, a major concern of the MHJS is to provide specialized marketplaces in which the System's prime commodity can be exchanged. Such a marketplace is called a Journal, and one speaks of "publishing" a document in a Journal. 4c1a

Many specialized Journals are anticipated, each attracting the attention of some segment of the the population. 4c1b

Some Journals which might arise are an 'AI Digest' in which work in the field of artificial intelligence is reported, a 'Journal of Graphics Protocol Development' in which systems programmers propose and discuss graphics protocols, a 'Resource News' in which Network service centers hawk their wares, and so forth. 4c1b1

Those users with interest in a particular Journal must be able to formally declare that interest, and, subject to appropriate access controls and accounting disciplines, place themselves in a position to be notified by the System whenever a document is published in that Journal. 4c1c

We shall call such a user a subscriber to that Journal.

A user can, of course, subscribe to any set of Journals he desires. 4c1c1

A particular document might be published in several Journals. We seek to permit the author of a document to conveniently keep tabs on where (i.e., in what Journals) his document's been published. 4c1d

It must be possible to assure that a potential contributor has the author's permission to publish the document, to limit the set of users who can publish in a particular Journal, and to force documents accepted for publication to meet certain requirements. 4c1e

A Journal's founder may, for a variety of reasons, wish to control the set of users who can publish documents in it. It's appropriate, for example, to permit only AI researchers to publish documents in the 'AI Digest'. 4c1e1

It may be necessary in many cases to have some assurance at publication time that a document will never be modified or deleted. 4c1e2

All such constraints, and others like them, must be individually assignable to a specific Journal, not inherent in the System's design. 4c1e3

Module Description 4c2

The System module which implements a Journal is called a Publisher. 4c2a

The Publisher's primary task is to catalog each document as it's contributed, and send a copy of the catalog entry (giving the article's author, title, date of publication, etc.) to each of its subscribers. 4c2b

The Publisher also screens potential contributors to determine whether they, and the document they propose to contribute, meet certain requirements. 4c2c

The contributor must have both write access to the Journal and distribute access to the document. 4c2c1

The Publisher may apply any other, additional tests it chooses before accepting a document for publication. 4c2c2

Several such additional tests suggest themselves, and

their application would lead to the following classes of Journals: 4c2c2a

PRIVATE and PUBLIC -- depending upon whether or not read access to the contributed document is to be restricted so as to include only Journal subscribers, 4c2c2a1

CLOSED or OPEN -- depending upon whether or not the right to publish in the Journal is granted exclusively to Journal subscribers, 4c2c2a2

PEMANENT and TRANSITORY -- depending upon whether or not published documents once published can ever be "unpublished" (or "unrecorded"), 4c2c2a3

The following hypothetical Journals make effective use of these three attributes in their various combinations: 4c2c2b

User Needs
(PUBLIC OPEN TRANSITORY) 4c2c2b1

This is an open forum -- anyone can contribute, anyone can read what's been contributed -- for the posting of resource wantads. Any interested users may subscribe. A service center might place a representative on the subscription list as a lookout for users whose needs they can satisfy (i.e., a lookout for potential customers). Articles are removed as they are responded to or when they become stale, 4c2c2b1a

Tenex Needs and Possibilites
(PUBLIC OPEN PERMANENT) 4c2c2b2

This is an open forum for proposal of feature additions to Tenex. Articles once published are forever published, so that a record of the system's development is preserved. Tenex programmers and selected administrative staff subscribe to this Journal, 4c2c2b2a

SRI Promotional & Transfer Opportunities
(PUBLIC CLOSED TRANSITORY) 4c2c2b3

This is the vechicle through which the SRI personnel department publicizes promotional opportunities to its staff members world-wide.

Any SRI employee can read posted notices (which are removed as positions are filled), but only personnel department representatives can post notices. One secretary from each department at each SRI site is on the subscription list. 4c2c2b3a

NIC AI Digest
(PUBLIC CLOSED PERMANENT) 4c2c2b4

This is a mechanism employed by NIC to keep interested members of the Network community informed about recent NIC acquisitions in the field of Artificial Intelligence. Each time an AI-related paper is received at the NIC, an abstract of it is published in this Journal. Anyone can read what's been published, but only NIC staff can contribute to the Journal. Because articles are never removed from the Journal, it becomes a valuable bibliographic source for AI researchers. 4c2c2b4a

A user's on-line mailbox
(PRIVATE OPEN TRANSITORY) 4c2c2b5

A user's on-line mailbox might be implemented as a Journal of this type, assuming he only received recorded mail. 4c2c2b5a

Tenex Bugs
(PRIVATE OPEN PERMANENT) 4c2c2b6

This is an open forum for the reporting of NLS bugs. Like 'Tenex Needs and Possibilities', it provides a permanent record of system growth, but unlike 'Needs and Possibilities', though anyone can contribute to it, only its subscribers (the Tenex programming staff) can read what's been published, to avoid the possibility of reports of potentially dangerous bugs being abused by maliscious users. 4c2c2b6a

A user's on-line filing cabinet
(PRIVATE CLOSED TRANSITORY) 4c2c2b7

The user publishes anything of interest to him -- important on-line mail, articles he's culled from outside sources, self-generated work reminders -- in his own, private Journal. Only he can publish in it or read its contents. He

is free to weed out previously-published articles based upon any criteria he finds appropriate, 4c2c2b7a

ARPA Intercom
(PRIVATE CLOSED PERMANENT) 4c2c2b8

This is a diary of ARPA, inter-office communication. Only ARPA staff members can publish in it or read its contents. The Journal represents a permanent record of office exchanges, 4c2c2b8a

In the same way that the System maintains a list of Recorders from whom a document can be obtained, so it maintains a list of Publishers in whose Journals the document has been published, 4c2d

NUMBER VENDOR 4d

The Need 4d1

As already described, each document recorded within the System is assigned a global handle called a document identifier, or DID. A mechanism must be provided by which DIDs can be orderly assigned and their status kept track of, 4d1a

For reasons of efficiency and reliability, it's highly desireable to provide a variety of sources from which DIDs can be obtained. Each additional source, assuming it resides on a different host, increases the probability of a user's being able to obtain a DID when he wants it, as well as reducing the overhead of obtaining it (by placing the source closer to him), 4d1b

Module Description 4d2

The System module which facilities the orderly assignment of DIDs is called a Number Vendor, 4d2a

Any number of Number Vendors may exist simultaneously within the System, and each, at any point in time, owns some subset of the universe of DIDs, from which it can satisfy user requests, 4d2b

The Number Vendor implements a primitive by which a user can obtain (i.e., be assigned) a block of DIDs for his use. A Number Vendor may only assign DIDs that it itself has been assigned by another Number Vendor, unless it's the "root"

Number Vendor which, when the System was created, found itself in possession of the entire name space, 4d2c

Other primitives are defined for returning an assigned DID to the Number Vendor (effectively unassigning it), marking a DID used (i.e., irrevocably associated with a document), verifying or changing a user's ownership of a DID, etc. 4d2d

One strategy which is available to the System as a whole is to station several Number Vendors throughout the System, each with responsibility for servicing its segment of the user population, and each replenishing its DID supply from the root Number Vendor when it nears bottom. This strategy permits a form of DID assignment which is both efficient and insensitive to the host failures which periodically make the root Number Vendor inaccessible. 4d2e

CATALOGER 4e

The Need 4e1

Many documents will be generated within the System, each with a unique DID. The DID is in itself sufficient information to permit the user to retrieve the contents of the document. A user could, therefore, in theory, discover every document in the System by simply trying each DID in turn at a nearby Recorder (of course, he could only read those to which he had been granted access). 4e1a

Such an approach is somewhat unsatisfying for the user. What's needed, of course, is a data base, called a catalog, which describes a selected subset of the documents recorded within the System. 4e1b

Each entry in the catalog will contain such information as the document's author (i.e., who placed it in the catalog), a title, the date of entry, and so forth. Collectively, this information is known as a citation. 4e1b1

Many catalogs will exist within the System, each with its own algorithm by which documents are included in it. 4e1c

Users must be permitted to search the catalog by a variety of algorithms. 4e1d

Access to the catalog must be controlled. The rights to interrogate the catalog, and to add to or delete a document from it must be independently assignable. 4e1e

Module Description 4e2

The System module responsible for maintaining a catalog is called a Cataloger. 4e2a

The Cataloger implements primitives by which other modules can add to or delete citations from the catalog, retrieve the citation for a specified document, and search the catalog by a variety of algorithms. 4e2b

The exercise of any primitive by a particular module is subject to access controls imposed by the Cataloger. 4e2c

Associated wth the catalog are lists of user to whom read, write, delete, and controlling access to the catalog are respectively granted. 4e2c1

A user must have read access to the catalog to invoke the retrieval or search primitive, delete access to invoke the delete primitive, and so forth. 4e2c1a

A user with controlling access to the catalog is permitted to change (by deleting from or adding users to) any of the catalog's access lists. 4e2c1b

The primitives by which a module actually effects such changes are not implemented by the Cataloger, but rather by a module called the Registrar described below. 4e2c2

REGISTRAR 4f

The Need 4f1

Keeping Track of Users and Modules 4f1a

Each module in the System must regularly apply a variety of access controls to properly constrain the use of its primitives. 4f1a1

To implement such controls, both human users and system modules are assigned names with which a password is then associated. 4f1a1a

The module then requires that the correct password be presented before it attempts to execute the primitive on behalf of the indicated user (that is, before the module will assume the user is who he says he is). Once the requestor's identity has been established,

the module can confidently make the necessary access checks, 4f1a1b

Other information (i.e., besides his password) about each user (or module) must also be maintained. A user's mailing address and delivery mode, for example, must be known to the System before mail from one user can be delivered to another. And similarly, a module's locatioon and access method must be known to the System for one module to contact another. 4f1a2

Mechanisms must be provided by which passwords can be verified, individual pieces of information about a particular user retrieved or modified, new users defined and old ones deleted, etc., and the use of all such mechanisms must be appropriately controlled. 4f1a3

Groups of Users and Modules 4f1b

Besides dealing with individual users and modules, the System must just as frequently deal with groups of users or modules. 4f1b1

The retrieve access list for a file is an example of such a list, and it must be consulted by the Storage Manager every time a user attempts to retrieve the file. 4f1b1a

The list of all Registrars within the System is representative of another class of lists which is usefully maintained by the System. Some primitives are only legally exercised by Registrars, and this particular kind of access control is then implementable by simply verifying the requesting user's memberhip in the Registrars group before honoring the primitive. 4f1b1b

Mechanisms must be provided by which a user's membership in a particular group can be established, names added to, deleted from, or replaced in the list, etc. 4f1b2

For reasons of efficiency and reliability, it's highly desireable to permit an arbitrary number of physical copies of subsets of this large data base to exist simultaneously within the System, since it will be the most frequently consulted data base in the System. 4f1c

Module Description 4f2

The System module responsible for maintaining information about users, System modules, and groups of users/modules is called a Registrar. 4f2a

IDENTS 4f2b

The Registrar trades in commodities called participants, each of which is designated by a unique ident. 4f2b1

A participant can be either an individual human user, an individual System module, or a group of users, modules or groups. 4f2b1a

For the most part, participants are treated identically by the System in general and the Registrar in particular, and a file's read access list, for example, may contain the idents of either human users or program modules, or a mixture of both. 4f2b1b

Idents are global handles on a participant, and therefore unique within the System. 4f2b2

The Registrar implements primitives which other System modules use to create and delete idents, retrieve or modify their memberships (in the case of group idents), verify an ident/password combination, and retrieve or modify the various other pieces of information which the Registrar may maintain for participants. 4f2b3

IDENT ACCESS CONTROL 4f2c

The exercise of any ident-manipulative primitive by a particular module is subject to ident-specific access controls imposed by the Registrar. 4f2c1

Associated with each ident are lists of users to whom read, write, append, delete, and controlling access to that ident are respectively granted. 4f2c2

A user must have read access to an ident to invoke any of the primitives which retrieve information about the designated paticipant, delete access to invoke the delete primitive, write access to modify information about the paticipant, and append access to add idents to the ident's membership list (assuming the ident designates a group). 4f2c2a

A user with controlling access to an ident is

permitted to change (by deleting from or adding idents to) any of the ident's access lists, 4f2c2b

The Registrar implements primitives by which any of the ident's access lists can be retrieved or modified, 4f2c3

IDENT NETWORKS 4f2d

Any number of Registrars may exist simultaneously within the System, each managing a COPY of some SUBSET of the data base, 4f2d1

An ident can be known to an arbitrary number of Registrars, and that particular set of Registrars is called the ident's domain, 4f2d2

Each Registrar in the ident's domain maintains a copy of all information pertinent to that ident, and can thus provide any subset of it to any module that requests it, 4f2d3

Modifications to the entry, necessarily requested of a particular Registrar, are relayed to all other Registrars affected. The various copies of the ident entry are thereby maintained consistent, 4f2d4

LOCKING 4f2e

One particular Registrar in the ident's domain is singled out as the ident's home, and it is at that Registrar that race conditions are resolved, 4f2e1

The Registrar implements a set of locking primitives by which he and other Registrars can gain appropriate access to an ident in preparation for manipulating it. An ident can be locked in such a way that just modifications, or both retrievals and modificatons to the ident are prohibited throughout its domain until the ident is unlocked, 4f2e2

The Registrar implements primitives by which an ident's home can be retrieved or modified, 4f2e3

THE REGISTRAR'S CENTRAL ROLE IN THE MHJS 4f2f

The Registrar turns out to be the workhorse of the MHJS. The importance of the Registrar to the whole System is so great that it seems worthwhile to explicitly state that fact here, 4f2f1

The number of ways in which the Registrar is employed by other modules is probably not evident to the reader of the current document, but is very evident in a more detailed design document (see -- 23143,1). 4f2f2

The Registrar's central role was not so much designed in as discovered. The discovered facts were: 4f2f3

(1) Virtually every class of System module must deal with incidental data bases which are lists of user/program names. Access lists are the most common data base of this type; others are the list of users who've read a document, the list of publishers who've published a document, a publisher's subscriber list, the list of all Registrars within the System, etc. Primitives must be provided by each module for retrieving and modifying these data bases. 4f2f3a

(2) System modules can be relieved of a significant burden by providing a specialized module whose function is to provide the primitives required to manipulate these data bases. 4f2f3b

(3) Once responsibility for these user lists is given to the Registrar, the lists become accessible from any one of an arbitrarily large set of Registrars (the group ident's domain), since the Registrar already implements the required broadcast facility. Since the list of users who've read a document and the list of publishers who've published it, for example, are universally available, an author or contributor can conveniently and automatically keep tabs on his document. In the same way, the list of Recorders who have copies of a specified document on hand is readily available to every user. 4f2f3c

(4) Since the existence of a document's read access list (for example) implies the existence of the document itself, whether or not a document exists can be determined by consulting the nearest Registrar. 4f2f3d

(5) Race conditions associated with the creation of a document (e.g., two users attempting to create a document with the same DID simultaneously at two different Recorders), for example, can be arbitrated by simply implementing Recorders such that they lock all of the various group idents associated with the document before preceding with its creation. 4f2f3e

Description of a Multi-Host Journal System

(J23144) 30-MAY-74 09:25; Title: Author(s): James E. (Jim) White/JEW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JEW; Origin: (WHITE, MHJSPAPER.NLS;33,), 29-MAY-74 19:10 JEW ;####;

KIRK 29-MAY-74 11:14 23145

Bug with jump to next on the last statement in a file

It acts like it's loading the file (SYSTEM, JOBPMF,;100003)
and then says "file numbers do not match in pushsrring" 1

KIRK 29-MAY-74 11:14 23145

Bug with jump to next on the last statement in a file

(J23145) 29-MAY-74 11:14; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /BUGS([ACTION]) KEV([ACTION]) ; Sub-Collections:
SRI-ARC BUGS; Clerk: KIRK;

Outline for NIC Final Report June 1974: MDK

Outline for NIC Final Report June 1974: MDK 1

I. Purpose and Scope of the NIC 2

A. History
B. Overall Objectives
C. Approach 2a

II. Services 3

A. On-Line Computer Services
B. Off-Line Documentation Services 3a

III. Difficulties 4

A. Management
B. Technical 4a

IV. Unfulfilled Needs 5

A. Network Resource Sharing
B. Selective Dissemination of Bibliographic Information 5a

V. Future Possibilities 6

A. Evolutionary Information Center
B. Distributed Information Center
C. Message Center 6a

MDK 29-MAY-74 08:09 23146

Outline for NIC Final Report June 1974: MDK

(J23146) 29-MAY-74 08:09; Title: Author(s): Michael D. Kudlick/MDK;
Distribution: /DVN; Sub-Collections: SRI-ARC; Clerk: MDK;
Origin: <KUDLICK>NIC.NLS;3, 29-MAY-74 08:07 MDK ;

DVN 29-MAY-74 08:43 23147

Keep Quarters Short

I checked your file (lee,fere,). It looks hopeful for the final report. The section for the QMR should not be more than one screenful. 1

DVN 29-MAY-74 08:43 23147

Keep Quarters Short

(J23147) 29-MAY-74 08:43; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /SRL([ACTION]) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN;

DVN 29-MAY-74 08:52 23148

Miscellaneous Nothing

When I called for Miscellaneous as an option in copy directory just now it showed for each file that it had no miscelaneous information, 1

DVN 29-MAY-74 08:52 23148

Miscellaneous Nothing

(J23148) 29-MAY-74 08:52; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /NEWNLS([ACTION]) ; Sub-Collections: SRI-ARC NEWNLS; Clerk: DVN;

DCE 29-MAY-74 09:03 23149

Bug in Output Processor for 23143

Tried this morning to do Output to Printer on (GJOURNAL, 23143, 0;wyn) -- twice -- and both times it bombed out partway through with:

```
Illegal Instructrion OPRTXT =  17152
  at [NLSLAN]ZRESE  =  413206
.....etc, five or so lines of noise (to me),
```

1

DCE 29-MAY-74 09:03 23149

Bug in Output Processor for 23143

(J23149) 29-MAY-74 09:03; Title: Author(s): Douglas C. Engelbart/DCE; Distribution: /FDBK([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: DCE;

DCE 29-MAY-74 09:11 23150

Property-Control process to have SRI review

Jim, Martin: Learned yesterday in ISE Lab Managers' meeting that there will be some outside group reviewing SRI's property control procedures, apparently in a very detailed way, between 3 Jun and 31 Jul. I gather that many items have been getting 'lost'. Bart asks each lab/center to get itself together on this count.

Jim, please give me some feedback about responsibility as currently being handled, a residue of your previous 'whole operations' roles; and suggestion for handling after we finish re-formulating responsibilies; and, who should be responsible for accommodating the above review. 1

DCE 29-MAY-74 09:11 23150

Property-Control process to have SRI review

(J23150) 29-MAY-74 09:11; Title: Author(s): Douglas C. Engelbart/DCE; Distribution: /JCN([ACTION]) MEH([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: DCE;

DCE 29-MAY-74 10:02 23152

Paychecks held for timecard arrival at Division Office

Everybody has a responsibility here

DCE 29-MAY-74 10:02 23152

Paychecks held for timecard arrival at Division Office

Timecards being processed in a timely fashion allow the Institute bookeeping that produces our paychecks. Tardy time-card processing is a continual strain in the ISE Division office -- each Lab/Center is supposed to have its cards and summaries handed in by noon on each Friday, and it often is much later (like 4). 1

The directors of ISE Labs and Centers unanimously agreed on the scheme that hereafter, on payday, the ISE Divison Office will release no checks to a Lab or Center until it has submitted its time-card material in proper form. 2

The Center secretary responsible for doing the tabulations on the cards, and the managers who have to initial them, before they can be sent to the Division Office, need a certain amount of time to accomplish these tasks. 3

Therefore, the rest of us must get our cards to them in time -- and must respond appropriately when they announce their deadlines. 4

Stand by for a series of announcements, as we learn how to work with deadlines. 5

DCE 29-MAY-74 10:02 23152

Paychecks held for timecard arrival at Division Office

(J23152
) 29-MAY-74 10:02; Title: Author(s): Douglas C. Engelbart/DCE;
Distribution: /BC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk:
DCE;

CURRENT CONTENT ANALYZER GLITCHES/BUGS

CURRENT CONTENT ANALYZER GLITCHES/BUGS 1

In TNLS, simple typed-in patterns are compiled but not instituted. When instituted with "INSTITUTE" command and invoked, they have no effect (all statements pass). 1a

In both TNLS and DNLS, short patterns cannot be deleted from the program buffer with "DELETE". Error message: "Invalid char in identifier". 1b

JDH 29-MAY-74 10:14 23153

CURRENT CONTENT ANALYZER GLITCHES/BUGS

(J23153) 29-MAY-74 10:14; Title: Author(s): J. D. Hopper/JDH;
Distribution: /KEV([ACTION]) FDBK([INFO-ONLY]) ;
Sub-Collections: SRI-ARC; Clerk: JDH;

SRL 29-MAY-74 13:15 23156

Which edges to shorten on quarters

Should the thing on feedback for the QMR emphasize results and opinions or stick mainly to a description or equal parts of both? 1

SRL 29-MAY-74 13:15 23156

Which edges to shorten on quarters

(J23156) 29-MAY-74 13:15; Title: Author(s): Susan R. Lee/SRL;
Distribution: /DVN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
SRL;

KIRK 29-MAY-74 17:01 23157

How to waste time, cpu and paper with indenting off

About 5 times a week, I have to reprint something because I had viewspecs B, and 1 or g indenting off and plex or branch only on. It looks indented in DNLS and TNLS, but is not indented when printed. This anomaly is impossible for me to get used to. There are three solutions that I know of. 1) make output quickprint work like the current system. 2) Change the current system by adding another viewspec for "left adjusted" independant of 1 and g, 3) Make 1 and g automatically mean "left adjusted" and capital B ALWAYS mean ALL indenting off. Number 2 seems like the way to go, but I don't see any one doing it. I would prefer 3 to the way it currently works if it would be easier to do than 2. But somebody please do something. I find "left adjustment" vital in working with highly structured files.

1

KIRK 29-MAY-74 17:01 23157

How to waste time, cpu and paper with indenting off

(J23157) 29-MAY-74 17:01; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /BUGS([ACTION]) KEV([INFO-ONLY]) ;
Sub-Collections: SRI-ARC BUGS; Clerk: KIRK;