DAV 22-OCT-75 14:55 26731

WUC loading bug

Your new version of WUC get two UNDEFINED GLOBALS when starting up NLS, (Apparently xprograms,wuc has been INCLUDED in my useroptions,) This is very confusing to the NLS user. 1

DAV 22-OCT-75 14:55 26731

WUC loading bug

(J26731) 22-OCT-75 14:55;;;; Title: Author(s): David C. Smith/DAV; Distribution: /KIRK([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: DAV;

26731 Distribution
Kirk E. Kelley,

The Distributed Programming System
Inducement to Share Resources Within a Computer Network

22-OCT-75

James E. White
Augmentation Research Center

Stanford Research Institute
Menlo Park, California 94025

The Distributed Programming System (DPS) is a software framework that extends the local programming environment to embrace processes in other hosts within a resource sharing computer network, thereby easing the task of building distributed systems and encouraging the sharing of resources.

SECOND DRAFT 23 OCT 75 8:10PM The Distributed Programming System

This is the second draft of the DPS paper we plan to submit for publication in one of the technical journals. My thanks to RWW, RLL, BEV, JAKE, and JBP who read through the first draft (26513,) and offered suggestions. The paper has been considerably reorganized, with the body of the paper shortened to make the key arguments more forceful (we hope), and secondary material relegated to appendices. Again, I would greatly appreciate the suggestions/corrections of anyone within ARC willing to offer them. Comments received by WED 29-OCT will be incorporated into the final version. An Output-processed version of the paper already exists as the file DPSPAPER.PRT in directory <WHITE> at ISIC and OFFICE-1, and in directory <JWHITE> at BBNB; copying this file to <ARCPRINTER> will get you formatted hardcopy that you can mark up. The figures alluded to in the paper do not yet exist; use your imagination.

*** SECOND DRAFT *** 1

THE PAPER 2

Resource Sharing, the ARPA Network's Goal 2a

A major goal of the now international, packet-switched computer network (the ARPA Network) constructed by the Advanced Research Projects Agency is to usefully interconnect geographically distributed hardware, software, and people resources [1]. Achieving this goal requires the design and implementation of various levels of support software within each "host" computer. This paper outlines an alternative to the approach that builders of such software have been taking since work in this area began in 1970, and suggests a strategy for modeling distributed systems within any large computer network. 2a1

Function-Oriented Protocols, a Means to the End 2b

The current ARPA Network software approach to facilitating resource sharing has been detailed elsewhere in the literature [2, 3, 4]. Briefly, it consists of defining first a Host-Host Protocol by which host operating systems cooperate to support a network-wide inter-process communication service (NIS), and then various function-oriented protocols (FOPs) by which resident "server processes" deliver specific services to "user processes" via NIS. 2b1

The current Host-Host Protocol has been in service since 1970 at, now, more than 75 host installations. Since its initial design and implementation, a variety of deficiencies have been recognized, and several alternative protocols suggested [5, 6]. Although the author recognizes the existence of such deficiencies and that their removal would enhance Network resource sharing, they are not the subject of the present paper, which assumes the existence of some form of NIS but focuses attention upon higher-level, FOP design issues. 2b2

Experience With and Limitations of Hands-On Resource Sharing 2c

The oldest and still by far the most heavily used FOP is the Telecommunications Network protocol (TELNET) [7], which effectively attaches a user's terminal on one host to an interactive time-sharing system on another, and allows him to interact with that system as if he were one of its local users. 2c1

As depicted in Figure 1, TELNET specifies the means by which a user process monitoring the user's terminal and a server process with access to the target time-sharing system are interconnected via an NIS communication "channel". It also legislates a standard character set in which the user's commands and the system's responses are to be represented in transmission between hosts. The syntax and semantics of these interchanges, on the other hand, vary from one host to another, and are unregulated by the protocol; user and server processes simply shuttle characters between the human user and the target system. 2c2

Although the "hands-on" use of remote resources that TELNET makes possible is a natural and highly visible form of resource sharing, it has several limitations that severely limit its long-term utility: 2c3

1) it forces upon the user all of the trappings of the host system that happens to contain the target resource.

   To apply a resource to the solution of his particular problem, the user must exchange the familiar working environment provided by his local system for an alien one with its own peculiar command language discipline and system structure. Hands-on resource sharing thus fails to provide the user with the kind of organized and consistent workshop he requires to work effectively [8].

2) it provides no basis for bootstrapping new composite resources from existing ones.

   Because the network access discipline imposed by each resource is its own human-engineered command language, rather than a machine-oriented communication protocol, it is virtually impossible for one resource to programatically draw upon the services of others. Hands-on resource sharing thus encourages an environment in which existing resources cannot be used as building blocks to construct new, more powerful ones.

These limitations, inherent in hands-on resource sharing, are removed by a protocol that simplifies and standardizes the dialog between user and server. Given such a protocol, the various remote resources upon which a user might wish to draw can indeed by made to appear as a single, coherent workshop by interposing between him and them a command language interpreter that transforms commands into the appropriate protocol utterances [9]. The construction of composite resources also becomes feasible, since the network interface to each resource is sufficiently simple that the user's command formulation expertise is no longer required and a program can be substituted for him. 2c4

Standardizing the Dialog in Specific Application Areas 2d

After the TELNET protocol had been designed and widely implemented within the ARPA Network, work began on a family of FOPs that each seek to facilitate "program-controlled resource sharing" by standardizing dialog in a particular application area. While TELNET dictates only the manner in which user and server are interconnected via NIS and the character set in which the two processes communicate once connected, each of these FOPs specifies in addition the syntax and semantics of the "commands" and command "responses" that comprise their dialog. 2d1

Protocols within this family necessarily differ in substance: each contains its own application-specific command set. Thus, the File Transfer Protocol (FTP) [10] specifies commands for manipulating files, while the Remote Job Entry Protocol (RJE) [11] defines commands for manipulating batch jobs. Protocols throughout the family are, however, similar in form, each successive family member having inherited the physical features of its predecessors. Thus FTP and RJE enforce the same conventions for formulating commands and responses. 2d2

This common "command/response discipline" (CRD) requires that commands and responses have the following respective formats: 2d3

```
name   <SP> parameter <CRLF>
number <SP> text      <CRLF>
```

Each command invoked by the user process is identified by NAME and allowed a single PARAMETER. Each response generated by the server process contains a three-digit decimal response NUMBER (to be interpreted by the user process) and explanatary TEXT (for presentation to a human user). Response numbers are assigned in such a way that, for example, positive and negative acknowledgments can be easily distinguished by the user process. 2d4

FTP contains, among others, the following commands (each listed with one of its possible responses) for retrieving, appending to, replacing, and deleting files within the server's file system, respectively: 2d5

```
RETR <SP> filename <CRLF>   250 <SP> Beginning transfer,
<CRLF>
APPE <SP> filename <CRLF>   400 <SP> Not implemented,
<CRLF>
STOR <SP> filename <CRLF>   453 <SP> Directory overflow,
<CRLF>
DELE <SP> filename <CRLF>   450 <SP> File not found,
<CRLF>
```

The first three commands serve to initiate the transfer of a file from one host to another; the transfer itself occurs on a separate NIS channel and is governed by what amounts to a separate protocol. 2d6

Multi-parameter operations must be implemented as sequences of single-parameter commands. Thus two commands are required to rename a file: 2d7

```
RNFR <SP> oldname  <CRLF>   200 <SP> Next parameter,
<CRLF>
RNTO <SP> newname  <CRLF>   253 <SP> File renamed,
<CRLF>
```

Factoring Out the Common Command/Response Discipline 2e

That FTP, RJE, and the other FOPs within this protocol family share a single CRD is a fact not formally recognized within the protocol literature, and each new protocol document describes it in detail, as if for the first time. Nowhere is the CRD codified in isolation from the various contexts in which it finds use, being viewed as a necessary but relatively unimportant facet of each FOP. The CRD has thus gone unrecognized as the important, application-independent protocol that it is. 2e1

This oversight has had two important negative effects upon the growth of resource sharing within the ARPA network. First, it has allowed the CRD to remain crude. As already noted, operations that require more than a single parameter are consistently implemented as two or more separate commands, each of which requires a response and thus incurs the overhead of a full round-trip network delay. Furthermore, there are no standards for encoding types of parameters other than character strings. Neither is there provision for returning results in a command response. Had the designers of the first FOP anticipated that it would become the prototype for a whole family of protocols, they would have designed a more flexible and efficient CRD. 2e2

The second effect of the oversight was to place upon the applications programmer the burden of implementing the CRD module, i.e. the code that formats outgoing commands, understands the details of NIS, and parses incoming responses. Rather than enabled to address remote processes at the level: 2e3

"invoke function DELE with argument TEXTFILE at host X"

the applications programmer is given only NIS as a foundation upon which to build, and required to construct the higher-level CRD software himself, which he invariably does, in every program he writes. The applications programmer is thus deterred from using remote resources by the amount of specialized knowledge and software that must first be acquired and built, respectively. 2e4

If, on the other hand, the CRD were formalized as a separate protocol, its use in subsequent FOPs could rightly be anticipated by the systems programmer, and a single CRD module constructed for use throughout a host installation (in the worst case, one implementation per programming language per host might be required). This module could then be placed in a library and link loaded (for example) into each new applications program, thereby greatly simplifying the use of remote resources. 2e5

As the importance of this module became evident, additional tasks would gradually be assigned to it, including, for example, those of automatically substituting for NIS a more efficient, intra-host form of inter-process communication whenever the remote process resided on the local host; automatically converting parameters between their internal and CRD-imposed formats; and handling error conditions. The module would thus provide an increasingly powerful, application-independent "network run-time environment" (NRTE), offering the applications programmer a variety of network programming conveniences. 2e6

The thesis of the present paper is that one of the keys to facilitating network resource sharing lies in isolating as a separate Protocol the CRD common to a large class of applications protocols; developing the Protocol to make it flexible and efficient; and constructing within each host a NRTE that by means of the Protocol provides the applications programmer with easy and high-level access to remote resources. 2e7

What a Flexible Command/Response Protocol Might Be Like 2f

Having argued the value of a CRD protocol as the framework for a large class of applications protocols, and of a NRTE module as a high-level interface to remote processes, there remains the task of suggesting the form that each of these entities might take. 2f1

The CRD protocol must possess a number of characteristics, the first of which are those of the discipline that it replaces. The Protocol must therefore: 2f2

1) permit invocation of arbitrary, named commands implemented by the remote process, and

2) permit command outcomes to be reported in a way that aids both the program invoking the commmand and the human user under whose control it executes.

Second, the Protocol should remove the known deficiencies of its predecessor, that is: 2f3

3) allow an arbitrary number of parameters to be supplied as arguments to a single command,

4) permit commands to return parameters as results, as well as accept them as arguments, and

5) provide representations for a variety of parameter types, including but not limited to character strings.

And finally, the Protocol should provide whatever additional capabilities are required to facilitate construction of the more complex distributed systems whose creation one seeks by means of the Protocol to encourage. Although others may later be identified, the two capabilities below are recognized now to be important: 2f4

6) permit the server to invoke commands in the user process,

In the workshop environment alluded to earlier, for example, graphical text editors (as servers) must invoke commands within the command language interpreter to manipulate the user's display.

7) permit a process to accept two or more commands for concurrrent execution.

That same text editor may wish to permit the user to initiate with one command a long formatting operation and yet continue to issue additional, shorter commands before the first has been responded to.

These seven requirements are met by the following pair of protocol utterances or messages: 2f5

```
COMMAND  tid name    arguments
RESPONSE tid outcome results
```

described here in purely symbolic form (Appendix C explores one possible encoding in detail). The first message invokes the command whose NAME is specified using the ARGUMENTS provided. The second is issued in eventual response to the first and returns the OUTCOME and RESULTS of the completed command. Whenever OUTCOME indicates that a command has failed, the RESULTS must be an error number and diagnostic message. 2f6

There are several elements of the Protocol that are absent from the existing FOP command/response discipline. The first is the single bit of information, COMMANd or RESPONSE, that distinguishes one type of message from the other. In the existing discipline, this distinction is implicit, since commands and responses only flow from user to server and server to user, respectively. This bit, therefore, arises from requirement six above. 2f7

The second new element is a "transaction identifier" (TID), which appears first in the command message and is later echoed in the response message. The TID provides the means by which a response is associated with the proper command, especially when two or more commands are executed concurrently. This new element thus arises from requirement seven above. 2f8

The final new element is only implied by the message descriptions above, i.e. the ability to transmit an arbitrary number of parameters, of various types, with each command or response. This requirement is most economically and effectively met by defining a small set of primitive "data types" (e.g. booleans, integers, character strings) from which concrete parameters can be modeled, and a "transmission format" in which the parameters can be encoded. Defining the transmission format in such a way that parameters are fully typed enables the NRTE to decode incoming parameters on behalf of the applications program, as previously suggested. Appendix A suggests a set of data types suitable for a large class of applications; Appendix B defines some possible transmission formats. 2f9

What the Run-time Environment Might Be Like 2g

Once the Protocol has been specified, the systems programmer can begin designing a NRTE for use by applications programmers at his installation. His task is to hide both conceptual and programming details from the applications programmer by providing a high-level model of the network and the appropriate software interface. His goal, of course, is to provide a run-time environment that as nearly as possible makes remote resources as easy to use as local ones. 2g1

Since local resources usually take the form of resident and/or library subroutines or "procedures", the possibility of modeling remote commands as procedures immediately suggests itself. This approach is further strengthened by the similarity one notes between local procedures and the remote commands to which the Protocol provides access. Both carry out arbitrarily complex named operations on behalf of the requesting program or caller, are governed by parameters or arguments provided by the caller, and return to him additional parameters or results reflecting the outcome of the operation. 2g2

Modeling commands as procedures thus acknowledges the fact that in a network environment, a program must sometimes call subroutines in machines other than its own. It also makes the NRTE the basis for a distributed programming system (DPS) that very elegantly extends the local programming environment to embrace processes throughout the network. It further suggests even the possibility of modifying compilers to provide minor variants of their normal procedure calling constructs for addressing remote procedures (and to in such cases drop out calls to the appropriate NRTE primitives). 2g3

This Model of the network environment is so attractive that it warrants backing up a step and adopting it network-wide. Doing so requires but a few cosmetic changes to the Protocol: 2g4

```
CALL   tid procedure arguments
RETURN tid outcome  results
```

and at the same time suggests a name for it: the Procedure Call Protocol (PCP). 2g5

A variety of more substantive additions to the Protocol will suggest themselves as the Model is expanded and its potential for taming the network environment more fully exploited. A number of such extensions are suggested in Appendix D. 2g6

Limitations of the DPS Analogy 2h

Although modeling the network environment as an extension of the local programming environment has great potential for facilitating the work of the network applications programmer, the DPS analogy does have limits that must be kept firmly in view. 2h1

First, remote procedure calls will be much more expensive than local ones, and the programmer must not allow this difference to drift too far into his subconscious or his handiwork will become increasingly inefficient. Like virtual memory, DPS offers great convenience and therefore power in exchange for reasonable alertness to the possibilities of abuse. 2h2

Neither is the analogy intended to imply that a distributed system must have a single locus of control. Many distributed systems will contain largely autonomous processes that only occasionally interact with one another. Furthermore, since each runs on a separate processor, a process need not suspend its own execution when it calls a procedure in one of its neighbors. DPS merely suggests a way of modeling interactions between processes; it imposes no particular constraints upon their other activities. 2h3

Nor is the analogy meant to imply that processes can coerce into action or otherwise unfairly manipulate one another. There is really no substantive difference between the procedure call and command/response models in this regard; a file creation procedure, for example, will not hesitate to complain when directory space has been exhausted. 2h4

Finally, one must recognize that by no means all useful forms of inter-process communication are effectively modeled as procedure calls. The existing NIS must therefore remain directly available to the applications programmer for those situations in which NRTE-provided primitives simply will not do. 2h5

Summing Up 2i

Despite the opportunities for abuse noted above, a fully evolved DPS has great potential for stimulating the sharing of resources within a computer network. First, it would significantly reduce the cost of installing existing programs as network resources by allowing a network interface consistent with their internal organization; and by eliminating the need for the design, documentation, and implementation of specialized delivery protocols. Second, it would encourage the use of remote resources by eliminating the need for application-specific interface software, thereby making remote procedures as accessible to the programmer as local ones. And finally, it would encourage the construction of new resources designed expressly for remote access, because of the ease with which they could be offered and used within the network software marketplace. 2i1

Acknowledgments 2j

Many individuals within both SRI's Augmentation Research Center (ARC) and the larger ARPA Network community have contributed their time and ideas to the development of the DPS concept and the design of a prototype distributed programming system. The contributions of the following individuals are expressly acknowledged: Dick Watson, Jon Postel, Charles Irby, Ken Victor, Dave Maynard, and Larry Garlick of ARC; and Bob Thomas and Rick Schantz of Bolt, Beranek, and Newman, Inc. 2j1

The work reported here was supported by the Advanced Research Projects Agency of the Department of Defense, and by the Rome Air Development Center of the Air Force. 2j2

APPENDICES 3

Appendix A -- DPS Data Types 3a

The Protocol requires that every parameter or "data object" be represented using one of several primitive data types defined by the Model. The set of data types below is sufficient to conveniently model a large class of data objects, but since the need for additional data types will inevitably arise, the set must remain open-ended. Throughout the descriptions below, N is confined to the range [0, 2**15-1]: 3a1

LIST A list is an ordered sequence of N data objects called "elements". A LIST may contain other LISTs as elements, and can therefore be employed to construct arbitrarily complex composite data objects.

CHARSTR A character string is an ordered sequence of N ASCII characters, and conveniently models a variety of textual entities, from short user names to whole paragraphs of text.

BITSTR A bit string is an ordered sequence of N bits and, therefore, provides a means for representing arbitrary binary data (e.g. the contents of a word of memory).

INTEGER An integer is a fixed-point number in the range [-2**31, 2**31-1], and conveniently models various kinds of numerical data, including time intervals, distances, etc.

INDEX An index is an integer in the range [1, 2**15-1]. As its name and value range suggest, an INDEX can be used to address a particular bit or character within a string, or element within a list. INDEXes have other uses as well, including the modeling of handles or identifiers for open files, created processes, etc. Because of their restricted range, INDEXes are more compact in transmission than INTEGERs (see Appendix B).

BOOLEAN A boolean represents a single bit of information, and has either the value true or false.

EMPTY An empty is a valueless place holder within a LIST or parameter list.

Appendix B -- PCP Transmission Formats 3b

Parameters must be encoded in a standard transmission format before they can be sent from one process to another via the Protocol. An effective strategy is to define several formats and select the most appropriate one at run-time, adding a format negotiation mechanism to the Protocol. Format negotiation would be another responsibility of the NRTE and could thus be made completely invisible to the applications program. 3b1

Suggested below are two transmission formats. The first is a 36-bit binary format for use between 36-bit machines, the second an 8-bit binary, "universal" format for use between dissimilar machines. 3b2

PCPB36, For Use Between 36-Bit Machines 3b3

```
Bits  0-13 Unused (zero)
Bits 14-17 Data type
   EMPTY  =1  INTEGER=4  LIST=7
   BOOLEAN=2  BITSTR =5
   INDEX  =3  CHARSTR=6
Bits 18-20 Unused (zero)
Bits 21-35 Value or length N
   EMPTY    unused (zero)
   BOOLEAN  14 zero-bits + 1-bit value (TRUE=1/FALSE=0)
   INDEX    unsigned value
   INTEGER  unused (zero)
   BITSTR   unsigned bit count N
   CHARSTR  unsigned character count N
   LIST     unsigned element count N
Bits 36-   Value
   EMPTY    unused (nonexistent)
   BOOLEAN  unused (nonexistent)
   INDEX    unused (nonexistent)
   INTEGER  two's complement full-word value
   BITSTR   bit string + zero padding to word boundary
   CHARSTR  ASCII string + zero padding to word boundary
   LIST     element data objects
```

PCPB8, For Use Between Dissimilar Machines 3b4

```
Byte    0  Data type
   EMPTY  =1  INTEGER=4  LIST=7
   BOOLEAN=2  BITSTR =5
   INDEX  =3  CHARSTR=6
Bytes 1-   Value
```

EMPTY	unused (nonexistent)
BOOLEAN	7 zero-bits + 1-bit value (TRUE=1/FALSE=0)
INDEX	2-byte unsigned value
INTEGER	4-byte two's complement value
BITSTR	2-byte unsigned bit count N + bit string + zero padding to byte boundary
CHARSTR	2-byte unsigned character count N + ASCII string
LIST	2-byte element count N + element data objects

Appendix C -- Detailed Encoding of the Procedure Call Protocol 3c

Although the data types and transmission formats detailed in the previous appendices serve primarily as vehicles for representing the arguments and results of remote procedures, they can just as readily and effectively be employed to represent the commands and responses by which those parameters are transmitted. 3c1

Taking this approach, one might model each of the two Protocol messages as a DPS data object, specifically a LIST whose first element is an INDEX message type. The following concise statement of the Protocol then results: 3c2

```
LIST (CALL,   tid,  procedure, arguments)
      INDEX 1 INDEX CHARSTR    LIST
LIST (RETURN, tid,  outcome,   results)
      INDEX 2 INDEX BOOLEAN    LIST
```

with the RESULTS of an unsuccessful procedure represented as follows: 3c3

```
LIST (error, diagnostic)
      INDEX  CHARSTR
```

Appendix D -- Possible Extensions to the Model 3d

Introduction 3d1

The simple Protocol proposed in this paper in itself provides the basis for a major extension to the local programming environment: a host- and application-independent means of calling procedures anywhere within the network. But upon this foundation can be constructed a wide range of additional facilities that further enhance the distributed programming environment by standarding other common forms of inter-process interaction. Within this appendix are offered examples of the many Model extensions that are possible, along with the additions to the Protocol required to effect them.

Since PCP's CALL and RETURN messages already provide a mechanism for invoking arbitrary remote procedures, the Model extensions suggested below will be implemented whenever possible as procedures, rather than as additional messages, thus bootstrapping new Protocol functions from old. These special "system procedures" are called and implemented by NRTEs, rather than by the applications programs they serve. Invoked in the standard way by means of the CALL message, they are distinguished from applications procedures by an INDEX number in place of the usual CHARSTR procedure name (see Appendix C).

In descriptions of the calling sequences of these system procedures, the notation "[x]" is shorthand for "x or EMPTY".

Extension A -- Coroutines and Signals 3d2

As defined by the Model, a procedure call is a very simple, two-stage dialog in which the caller describes the operation it wishes performed and the callee, after performing the operation, describes its outcome. Although this simple dialog form is sufficient to conveniently implement a large class of distributed systems, more complex forms are sometimes required. The Model can readily be extended to admit a variety of more powerful dialog forms, of which the two described below are examples.

Effecting Coroutine Linkages

In conventional programming systems, the concept of "coroutines" is often introduced to permit caller and callee to exchange parameters (and control) any number of times before the callee returns. Coroutine linkages provide a means, for example, by which the callee can obtain help with a problem that it has encountered, or return the results of one sub-operation and obtain the arguments for the next.

Extending the Model to embrace coroutines requires adding to the Protocol described in Appendix C, a message that "transfers control" between caller and callee:

```
LIST (XFERCTRL, tid,    tocaller?, parameters)
      INDEX 3   INDEX  BOOLEAN    LIST
```

and specifies the identifier TID of the transaction (i.e. call) to which it pertains; the direction (TOCALLER?) of the control transfer, necessary to identify the name space from which the TID was assigned; and the PARAMETERS provided by caller or callee. Use of the XFERCTRL message is depicted in Figure 2.

Signalling

Sometimes a monolog is more appropriate than the dialog that a coroutine linkage initiates. The caller or callee might wish, for example, to report an event it has detected, or send large parameters piecemeal to minimize buffering requirements. In such cases, the initiating procedure requires no response from its partner and desires to retain control of the call.

The Model can be extended to support "signals" of this type by adding to the Protocol the following system procedure, which transmits parameters between caller and callee while retaining control of the call:

```
SGNLPROC (tid,  tocaller?, parameters)
           INDEX BOOLEAN    LIST
```

and which, like the XFERCTRL message described above, specifies the transaction identifier TID of the call to which it pertains, the direction (TOCALLER?) of the parameter transfer, and the PARAMETERS themselves.

Implementing signals as a system procedure rather than a message provides a crude form of flow control that prevents the receiving NRTE's buffers from being overrun. The signalling primitive that the NRTE makes available to the applications program, however, should initiate the call to SGNLPROC but not delay the user program by waiting for its return. Only when it signals a second time (as depicted in Figure 3) or releases control of the call should the NRTE make certain that the previous call to SGNLPROC has completed before proceeding.

Extension B -- Control Thread Communication 3d3

As in conventional programming systems, remotely-callable procedures within a distributed system will sometimes call upon others to carry out portions of their task. Each procedure along the "thread of control" resulting from such nested calls is, in a sense, acting under the authority of and is therefore responsible to, not just its immediate caller but also those procedures above it. To properly discharge this responsibility, a procedure must sometimes communicate with these "superior" procedures; e.g. to report an event that it has detected or caused, or to solicit help with a problem that has been encountered.

The Model can easily be extended to admit various forms of control thread communication. The two examples described below are analogous to the coroutine and signal facilities described in Extension A above (they support dialogs and monologs, respectively), but these additional facilities are always directed up the control thread, to superiors as a class (rather than specifically to the caller).

Soliciting Help with a Problem

Occasionally a procedure reaches a point in its execution beyond which it cannot proceed without external assistance. It might, for example, require additional resources, or further direction from the human user upon whose behalf it is executing. The procedure may have invested considerable real and/or processing time before reaching this impasse, all of which will be lost if it aborts.

To minimize such inefficiencies, the Model can be extended to permit a "stuck" procedure to solicit help from its superiors, simply by adding the following system procedure to the Protocol:

```
HELP (tid,    number,  information -> solution)
      INDEX  INDEX    any              any
```

which specifies the transaction identifier TID of the call to which it pertains (the direction of the control transfer being implicit), the NUMBER of the problem encountered, and arbitrary supplementary INFORMATION.

As depicted in Figure 4, the search for help begins with invocation of HELP in the caller's NRTE. If the caller understands the problem (i.e. recognizes its number) and is able to solve it, HELP simply returns the provided SOLUTION information. Otherwise, HELP calls itself recursively in the next process up the thread of control. The search terminates as soon as one of the stuck procedure's superiors responds positively, or when the end of the control thread is reached. In the latter case, each of the nested HELP procedures returns unsuccessfully in turn to indicate that the search failed.

Reporting an Event

A procedure sometimes witnesses or causes an event of which its superiors should be made aware: the start or completion of some major step in the procedure's execution, the reaching of a breakpoint, etc.

The Model can be extended to permit a procedure to notify its superiors of an arbitrary event, simply by adding the following system procedure to the Protocol:

```
NOTE (tid,  number, information)
      INDEX INDEX   any
```

which, like the HELP procedure described above, specifies the transaction identifier TID of the call to which it pertains, the NUMBER of the event being reported, and arbitrary supplementary INFORMATION.

As depicted in Figure 5, notification of the procedure's superiors begins with invocation of NOTE in the caller's process, and works its way recursively up the thread of control until the top is reached.

Implementation of the note function as a system procedure, rather than as a message, provides a crude form of flow control that prevents the receiving NRTE's buffers from being overrun. The note primitive that the NRTE makes available to the user program, however, should initiate the call to NOTE but not delay the applications program by waiting for its return. Only when it notes a second event or releases control of the call should the NRTE make certain that the previous call to NOTE has completed before proceeding.

Extension C -- Procedure Interruption and Abortion 3d4

In conventional systems, it is important to provide mechanisms for forcibly aborting, and perhaps suspending and resuming procedures that are called as the direct result of commands issued by a human user. The ability to abort a procedure is particularly important when the procedure has a significant execution time (e.g. a procedure that compiles a source file).

In distributed systems, the same needs exist. The Model can be extended to provide these facilities by adding the following three system procedures to the Protocol:

```
INTPROC (tid)
        INDEX
RSMPROC (tid)
        INDEX
ABRPROC (tid)
        INDEX
```

which interrupt, resume, and abort, respectively, the procedure whose transaction identifier TID is specified. As depicted in Figure 6, these procedures may only be invoked on behalf of the caller.

Extension D -- Data Stores 3d5

Most conventional systems maintain a variety of state information that they house in global variables or "data stores". These data stores are usually manipulated using primitives of the form: "replace the current contents of D with the value V" and "return the current value of D". These primitives are implemented either by specialized procedures or as language constructs; the latter approach offers the advantage of insuring uniform handling of all data stores within the system.

Those distributed systems that maintain remotely-accessible state information must also select a strategy for implementing the required data store access primitives. Although they can be implemented as specialized user procedures, a simple extension to the Model would provide a uniform run-time access mechanism. A suitably modified compiler could in turn exploit this run-time uniformity to provide the compile-time uniformity that conventional programming environments offer.

Reading and Writing a Data Store

If a data store be defined as a named data object modeled from DPS data types, the required access primitives are provided by adding the following pair of system procedures to the Protocol:

```
WRITSTOR (store,    value)
          CHARSTR   any
READSTOR (store -> value)
          CHARSTR   any
```

which specify and return the VALUE of the indicated data STORE, respectively.

Data stores may restrict the kinds of manipulation to which they are willing to submit: some may be read-only, some may accept values of only a certain data type, etc.

Manipulating an Element of a Data Store

Because they are modeled using DPS data types, the values of data stores can be arbitrarily complex, e.g. a LIST of LISTs. The programmer may sometimes wish to manipulate only a single element of the LIST; or, if the selected element is itself a LIST, he might wish to manipulate just one of its elements; and so on, to arbitrary depth.

WRITSTOR and READSTOR can be extended to optionally write or read a particular substructure within a composite data store:

```
WRITSTOR (store,  substructure,          value)
          CHARSTR [LIST (index, ...)] any
                          INDEX
READSTOR (store,  substructure -> value)
          CHARSTR (as above)      any
```

At each successive level of the data STORE's tree structure, the INDEX of the desired element is identified; the resulting list of indices identifies the SUBSTRUCTURE whose VALUE is to be replaced or returned.

Extension E -- Parameter List Masks 3d6

In conventional programming systems, the results of procedures are used in a variety of ways, depending upon the context of the calls made upon them. A result may, for example:

1) provide the basis for a branch decision within the calling program,
2) become an argument to a subsequent procedure call, or
3) be ignored and thus effectively discarded.

At run-time, the knowledge of a particular result's intended use usually lies solely within the calling program, which examines the result, passes it to a second procedure, or ignores it as it chooses.

In distributed systems, the results of remote procedures will be used in precisely the same ways. But here the transportation of results from callee to caller, being carried out by means of one or more inter-process messages, is an expensive operation, especially when the results are large.

If the Model were extended in such a way that the intended disposition of a procedure's results could be made known to the callee's process in advance, data movement could be reduced in cases 2 and 3 above. In case 2, provided both callees reside within the same process, the result could be held at its source and later locally supplied to the next procedure. In case 3, the result could be discarded at its source (perhaps not even computed), rather than sent and discarded at its destination.

Specifying Parameters Indirectly

Data stores (see Extension D) offer potential for eliminating the inefficiencies involved in case 2 above by providing a place within the callees' process where results generated by the first procedure can be held until required by the second.

The Model can be extended to permit data stores to be used in this way by allowing the caller of any procedure to provide "argument and result list masks" like the following:

LIST (store, ...)
[CHARSTR]

to specify the source or destination of each of the procedure's arguments and results, respectively. As depicted in Figure 7, a parameter list mask like that above would permit each parameter to be transmitted either directly, via the parameter list, or indirectly via a data STORE within the callee's process; each element of the mask specifies how the callee's process is to obtain or dispose of the corresponding parameter.

The Model can be extended as suggested above by modifying the Protocol to allow argument and result list masks as additional parameters of the CALL message. To supply the result of one procedure as an argument to another, the caller need only then appropriately set corresponding elements of the result list mask in the first call and the argument list mask in the second (the result list mask should be ignored if the procedure fails, and the error number and diagnostic message returned directly to the caller).

Discarding Results

The inefficiencies that result in case 3 above are conveniently eliminated by allowing the caller to identify via the result list mask (e.g. via a zero-length CHARSTR) that the result to be discarded will in fact be ignored and therefore need not be returned to the caller.

Extension F -- Scratch Data Stores 3d7

Although each applications program could maintain "scratch" data stores for use as suggested in Extension E above, a more economical approach is to extend the Model to permit such data stores to be created and deleted at run-time as necessary. Scratch data stores could then be maintained entirely by the NRTE, without any assistance from the applications program.

The necessary primitives are provided by adding the following pair of system procedures to the Protocol:

```
CRTSTOR (store,  value)
         CHARSTR any
DELSTOR (store)
         CHARSTR
```

which create (and assign an initial VALUE to) and delete a data STORE, respectively.

Extension G -- Packages

3d8

Large distributed systems are constructed by logically interconnecting autonomous processes in potentially complex ways. A user process P1, for example, might draw upon the resources of a second, service-vending process P2 that employs a third process P3 to maintain its accounting records. In such an arrangement, P1 would have no need or right to deal directly with P3. It might be desirable, therefore, to extend the DPS Model to permit access controls to be imposed upon individual resources.

Procedures (and data stores, if Extension D is made) are the building blocks of the distributed systems that DPS makes possible. Although an individual procedure or data store can be thought of as a resource, it is usually more practical and convenient to conceive of larger, composite resources comprised of a number of related procedures and data stores. A simple data base management module, containing procedures for creating, deleting, assigning values to, reading, and searching for data objects, exemplifies such composite resources. Although each procedure is useless in isolation, the whole family of procedures provides a meaningful service. Such composite resources or "packages" of logically related procedures and data stores might thus be the most reasonable object of access controls.

Access controls might be applied to packages by requiring that a process "open", and perhaps as a result obtain a handle for a remote package before it can access any of the procedures or data stores it contains. Whenever a process attempted to open a package, its right to do so could then be verified and the attempt aborted if necessary. Challenging the initial attempt to open the package would, of course, be less expensive than challenging every procedure call. The opening of a package would also provide a convenient time for package-dependent state information to be initialized.

The necessary primitives for opening and closing remote packages might be provided by adding to the Protocol the following pair of system procedures that, for efficiency, manipulate an arbitrary number of packages in a single transaction:

```
OPENPACK (names ->              pkhs)
          LIST (package, ...,) LIST (pkh, ...)
            CHARSTR              INDEX
CLOSPACK (pkhs)
          (as above)
```

The first of the two procedures opens and returns handles PKHS for the packages whose NAMES are specified; the second closes one or more packages and releases the handles PKHS previously obtained for them.

Having added the two primitives above, the Protocol must also be modified to require a package handle with each procedure and data store name in messages and system procedures. Thus, procedures and data stores would be designated throughout the Protocol by the following composite data object:

```
LIST (pkh,  store/procedure)
      INDEX CHARSTR
```

rather than by a simple procedure or data store name.

Note in passing that this scheme would make the package the domain over which procedure and data store names must be unique. Package names could also serve, in dialog and documentation, as convenient shorthands for more detailed descriptions of the procedures and data stores they contain.

Extension H -- Process Initialization 3d9

Before a program in one host can exploit the resources of another host, it must, in general, create a new process within that host. Creation of a process is a multi-step operation that involves:

1) establishing a channel to the remote host via NIS;
2) presenting credentials for purposes of billing and access to the host's file system;
3) identifying the program to be run (i.e. the resources that are sought);
4) winning the commitment of a processor to that program; and finally
5) having the program started.

The means by which these required steps are effected depends heavily upon the nature of the NIS environment in which the distributed system is constructed. Assuming that specification of accounting parameters and program name is not an intrinsic part of establishing the channel, this information must be explicitly transmitted by the creating process.

The Model is easily extended to incorporate the specification of the required startup information by adding the following system procedure to the Protocol:

```
INIT (program, credentials)
      CHARSTR  LIST (user,   password, account)
                     CHARSTR CHARSTR   CHARSTR
```

which identifies the user PROGRAM to be run, and provides the USER name, PASSWORD, and ACCOUNT of the local user upon whose authority the process is to be created.

Extension I -- Process Interconnection 3d10

The simplest distributed systems begin with a single process that creates one or more "inferior" processes whose resources it requires. One or more of these inferiors may in turn require other remote resources and so create inferiors of their own. This creative activity can proceed, in principle, to arbitrary depth. The distributed system that results is a tree structure whose nodes are processes and whose branches are NIS channels. Any particular process, however, is only aware of its parent and sons (i.e. the process that created it, and those it itself created). Because each process is so near-sighted, the radius within which it can access the resources of the tree is artificially small.

Because this limited sharing range is insufficient to implement many distributed systems, the Model might be extended to permit a process to "introduce" and thereby make known to one another any two processes it already knows (e.g. two of its sons, or its parent and son). Once introduced, the two processes would be free to invoke one another's procedures with the same freedom the introducing process enjoys, until it "separates" them.

The introduction of two processes requires little more than the formation of an NIS channel between them. This channel might be negotiated by the NRTE of the process performing the introduction, using the family of system procedures below:

```
ALOPORT (=> localport)
            LIST (host, portnumber)
                 INDEX INTEGER
CNNPORT (localport, remoteport)
         (as above) (as above)
DCNPORT (localport)
         (as above)
```

To create the channel, the introducing process allocates an NIS "port" in each target process by calling its ALOPORT procedure, as depicted in Figure 8. It then commands each of the target processes, via CNNPORT, to connect its port to the port allocated by the other. To later disconnect the two processes, the separating process simply invokes the DCNPORT procedure in each process, thereby dissolving the channel and releasing the associated ports.

The scenario above is only meant to be representative. The detailed calling sequences of the required procedures depend heavily upon NIS. The calling sequences required by the ARPA Network's NIS, for example, are only slightly more complicated than those above.

Extension J -- Logical Channels 3d11

In an environment in which all processes share a common NIS (the premise of Extension I above), the introduction of two processes necessitates only the formation of an NIS channel between them. In other, less homogeneous environments, however, the creation of such a channel is not always possible.

Suppose, as depicted in Figure 9, that processes P1 and P2 (in hosts H1 and H2, respectively) are interconnected within a distributed system by means of the NIS of a large computer network (e.g. the ARPA Network). Assume further that P2 attaches to the system another process P3 in a minicomputer M that although attached to H2 is not formally a part of the network. With this configuration, it is impossible for P2 to introduce processes P1 and P3 to one another by simply establishing an NIS channel between them, since they are not within the domain of a single inter-process communication facility.

One way of solving this problem is to extend the Model to embrace the notion of a composite or "logical channel" composed of two or more physical channels. A message transmitted by process P1 via the logical channel to Pn (n=3 in the example above) would be relayed over successive physical channels by intermediate processes P2 through Pn-1. As depicted in Figure 10, a logical channel would consist of a table entry maintained by the NRTE of each process P1 through Pn, plus the commitment of each NRTE to forward messages that arrive with a "routing code" that addresses the table entry. Each table entry would contain pointers to the two neighboring processes, as well as the routing code recognized by each; the message's routing code would be updated from the table entry before the message was forwarded.

To communicate a message to its distant neighbor, therefore, the source process (say P1) would transmit it via its NIS channel to P2, with a routing code addressing the appropriate table entry within P2. Upon receipt of the message, P2 would locate its table entry via the routing code, update the message with the routing code that P3 recognizes, and forward the message to P3. Eventually the message would reach its final destination, Pn. The generality of logical channels is thus not secured without cost, since each message must traverse at least two physical channels and be handled by all of the NRTEs along the way.

A logical channel of the sort described above could be suitably manipulated using the two system procedures described below:

```
CRTROUTE (mycode, modelcode -> yourcode)
          INDEX   [INDEX]      INDEX
DELROUTE (yourcode)
          INDEX
```

which create and delete, respectively, the table entry (within the called process) whose routing code YOURCODE is specified.

The simplest logical channel (n=3) is created by P2, which invokes CRTROUTE in both P1 and P3, specifying its routing code MYCODE for the logical channel, and receiving routing codes YOURCODE assigned by P1 and P2 in return; MODELCODE is EMPTY. More complicated logical channels (n>3) result when one or both of the introduced processes is already linked by a logical channel to the process performing the introduction. In this case, MODELCODE specifies the routing code for the table entry within the target process that represents the existing logical channel to be reproduced. The target process must then call CRTROUTE recursively to replicate the rest of the model channel.

The process Pi that creates the logical channel would be responsible for eventually dismantling it by invoking DELROUTE in Pi-1 and Pi+1, each of which would in turn propagate the call to its end of the channel.

Other Areas for Investigation 3d12

The Model extensions suggested in the preceeding sections are intended to be representative of the many that might usefully be made. Among the additional facilities that might eventually be provided by the NRTE by means of the Protocol are mechanisms for:

1) queuing procedure calls for long periods of time (e.g. days),
2) broadcasting requests to several processes,
3) subcontracting work to other processes without remaining a middleman,
4) supporting brief or infrequent inter-process exchanges with minimal startup overhead, and
5) effective error recovery and restart.

These and other areas must be explored, and the results of that exploration made a part of the coherent framework that DPS provides.

Appendix E -- An Implementation of the Run-time Environment 3e

Background 3e1

In July of 1974, the Augmentation Research Center (ARC) at SRI undertook the design of a DPS for use in the National Software Works (NSW) being constructed for the Air Force's Data Automation Agency (DAA), by the Advanced Research Projects Agency (ARPA) and Rome Air Development Center (RADC).

The work consisted of developing the DPS model; designing and documenting the Network protocol required to support it [12]; and designing, documenting, and implementing a NRTE for a particular machine [13, 14], specifically a PDP-10 running the Tenex operating system developed by Bolt, Beranek, and Newman, Inc [15]. Three design iterations were carried out during the course of the next twelve months, and the final design implemented on Tenex. The resulting system is a superset of that described in the body of this paper and Appendix D.

In this final appendix, the structure of the NRTE implemented by ARC for Tenex is briefly sketched (others are, of course, possible). In general, the design and implementation of a NRTE is a highly host-dependent task. This description, then, is provided as an example of how a network run-time environment might be structured.

The Tenex Operating System 3e2

The Tenex operating system supports the concurrent execution of a potentially large number of user jobs, each of which consists of one or more hierarchically-related processes called "forks". Each fork implements a PDP-10-like virtual machine with 256K of virtual memory.

Forks within a job are arranged in a tree structure. Each fork has exactly one immediately superior fork and zero or more immediately inferior forks. A fork can communicate with its inferiors by sharing memory and/or by means of asynchronous signals called "pseudo interrupts". A superior fork can also forcibly freeze, resume, or kill any of its inferiors.

Operating system services are made available to user forks by means of system calls, or "JSYSes" (Jump-to-SYStem's). Tenex permits a superior to trap any or all JSYSes invoked by any or all of its inferior forks, examine their arguments, and if desired even assume the task of implementing them [16].

Isolating the NRTE in a Separate Fork 3e3

As depicted in Figure 11, the NRTE is isolated in a separate "controlling fork" (CF), while the applications program executes in one or more immediately inferior "procesing forks" (PFs). Every DPS process on Tenex thus consists of at least two forks, one containing the NRTE and the other the applications program. The CF makes the services of the NRTE available to PFs by means of a single system call which it intercepts via the trap facility that Tenex offers. Upon this single, intercepted JSYS are multiplexed a family of "virtual VJSYSes" (VJSYSes) that constitute NRTE primitives.

This approach offers a number of advantages over others involving the link loading of the NRTE with the applications program and their execution in a single address space:

1) it makes DPS appear to the applications programmer to be part of the operating system,

   NRTE primitives adhere to standard Tenex conventions for passing parameters, indicating outcome, etc.

2) it makes the NRTE language-independent and thus permits a single implementation to be maintained,

   This implementation can be used by any applications programmer on any Tenex machine within the ARPA Network.

3) it protects the NRTE from the applications programmer and thus eases the debugging task,

4) it eliminates such potential conflicts between the NRTE and the applications program as inconsistent use of the pseudo interrupt system, and finally

5) it allows the NRTE to execute independently of the applications program, as it must do, for example, to effectively monitor the NIS channels attached to the process,

Giving the NRTE Access to the Applications Program 3e4

As the applications program must invoke primitives provided by the NRTE, so must the NRTE draw upon the resources of the applications program. The NRTE must, for example, be able to cause the appropriate procedure within the applications program to be dispatched whenever a CALL message arrives from an adjacent process. As a vehicle for such NRTE-driven interactions, the notion of a "virtual Jump-to USeR" (VJUSR), implemented by the PF and available to the CF, is introduced. The NRTE specifies the calling sequences of a family of VJUSRs and requires their implementation by the applications program.

Additional Services Provided by the NRTE 3e5

In the body of this paper it was suggested that the NRTE would gradually grow in capability to provide the applications programmer with a variety of network programming conveniences. Included among those provided by the initial Tenex NRTE are:

1) automatic selection of the most appropriate mode of inter-process communication,

   Currently, intra-job communication is carried out by means of shared memory; NIS channels are employed in all other cases.

2) automatic selection of the most appropriate transmission format for each adjacent process,

   Currently, the PCPB36 format (see Appendix B) is employed between Tenex machines, and the PCPB8 format used in all other situations. The applications program, however, always communicates with the NRTE using the PCPB36 format, the NRTE performing whatever format conversion is required,

3) the ability to combine two or more independently-written programs in a single process,

Thus separately-coded packages can be combined at run-time in a way that makes their compile-time independence invisible to other processes,

4) automatic scheduling of incoming procedure calls among those PFs executing the appropriate applications program,

5) primitives for creating and deleting PFs and for adding to and removing applications programs from the process' run-time configuration,

6) automatic creation and deleting of PFs in accordance with the demands of adjacent processes, and

7) primitives for intra-process communication and synchronization,

REFERENCES 4

1. Kahn, R. E. Resource-Sharing Computer Communications Networks. Proc. IEEE, Vol. 60, No. 11, 1397-1407 (Nov. 1972). 4a

2. Crocker, S. D., Heafner, J. F., Metcalfe, R. M. and Postel, J. B. Function-oriented Protocols for the ARPA Computer Network. AFIPS Conf. Proc., SJCC, Vol. 40, 271-9 (1972). 4b

3. Carr, C. S., Crocker, S. D. and Cerf, V. G. Host-Host Communication Protocol in the ARPA Network. AFIPS Conf. Proc., SJCC, Vol. 36, 589-597 (1970). 4c

4. Mc Kenzie, A. A. Host/Host Protocol for the ARPA Network, Bolt Beranek and Newman Inc., Cambridge, Mass., Jan. 1972. (NIC 8246) 4d

5. Walden, D. C. A System for Interprocess Communication in a Resource Sharing Computer Network. Commun. ACM, Vol. 15, No. 4, 221-30 (Apr. 1972). 4e

6. Cerf, V. G. and Kahn, R. E. A Protocol for Packet Network Intercommunication, IEEE Trans. Commun., Vol. Com-22, No. 5, 637-48 (May 1974). 4f

7. TELNET Protocol Specification, Stanford Research Institute, Menlo Park, Calif., Aug. 1973. (NIC 18639) 4g

8. Engelbart, D. C., Watson, R. W. and Norton, J. C. The Augmented Knowledge Workshop. AFIPS Conf. Proc., NCC, Vol. xx, 9-21 (1973). 4h

9. CLI 4i

10. Neigus, N. J. File Transfer Protocol, RFC 542, Bolt Beranek and Newman, Inc., Cambridge, Mass., Jul. 1973. (NIC 17759) 4j

11. Bressler, R. D., Guida, R. and Mc Kenzie, A. A. Remote Job Entry Protocol, RFC 360, Dynamic Modeling Group, Massachusetts Inst. Technol., Cambridge, Mass., n.d. (NIC 12112) 4k

12. White, J. E. DPS-10 Version 2.5 Implementer's Guide, Augmentation Research Center, Stanford Research Inst., Menlo Park, Calif., Aug. 15, 1975. (NIC 26282) 4l

13. White, J. E. DPS-10 Version 2.5 Programmer's Guide, Augmentation Research Center, Stanford Research Inst., Menlo Park, Calif., Aug. 13, 1975. (NIC 26271) 4m

14. White, J. E. DPS-10 Version 2.5 Source Code, Augmentation Research Center, Stanford Research Inst., Menlo Park, Calif., Aug. 13, 1975. (NIC 26267) 4n

15. Bobrow, D. G., Burchfiel, J. D., Murphy, D. L. and Tomlinson, R. S. TENEX, a Paged Time Sharing System for the PDP-10, Commun. ACM, Vol. 15, No. 3, 135-43 (Mar. 1972) 4o

16. Thomas, R. H. JSYS Traps, A TENEX Mechanism for Encapsulating Processes. AFIPS Conf. Proc., NCC, Vol. xx, yyy-z (1975). 4p

FIGURES 5

26732 Distribution

Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Israel A. Torres, Jan H. Kremers, Susan K. Ocken, Raphael Rom, David C. Smith, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Ann Weinberg, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White

JBP 22-OCT-75 18:52 26733

line processor vs <ESC> character

at isic in display nls looking at the file nic-nls,runldr i got int a state where the line processor blew up, hitting system restet on the lp caused it to begin to refersh the screen again but it blew up part way thru,,, infinite loop, and fun to get out of, problem seems to be an escape character in the file, such things should not blow the lp !!! --jon, 1

JBP 22-OCT-75 18:52 26733

line processor vs <ESC> character

(J26733) 22-OCT-75 18:52;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

26733 Distribution
Special Jhb Feedback,

DRM3 23-OCT-75 11:23 26734

notice of reassignment

all personnel having any knowledge of the arpanet are here by
transfered to to the bindery!!!!! see you there? 1

notice of reassignment

(J26734) 23-OCT-75 11:23;;;; Title: Author(s): Donald R. McEntee/DRM3; Distribution: /FMM([ACTION]) FJS([ACTION]) TJC([ACTION]) OLP([ACTION]) DRM3([INFO-ONLY]) JBK([INFO-ONLY]) CDH([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: DRM3;

26734 Distribution
Frank M. Mirkay, Flynn J. Stubblefield, Tom J. Cook, Opal L. Power,
Donald R. McEntee, John B. Kemery, Chuck D. Hall,

DRM3 23-OCT-75 12:09 26735

WELCOME

HELLO, WE ARE TAKING THE TRAINING COURSE FROM SRI ON TNLS, REALLY LIKE THE SYSTEM! BE TALKING TO YOU LATER. 1

WELCOME

(J26735) 23-OCT-75 12:09;;;; Title: Author(s): Donald R. McEntee/DRM3; Distribution: /JPC([ACTION]) CDH([INFO-ONLY]) ; Sub-Collections: NIC; Clerk: DRM3;

26735 Distribution
Joseph P. Cavano, Chuck D. Hall,

Message Reformatting

Ed: Do try the message subsystem on your Journal Messages, I think it will format them enough for your needs right now, I don't know where my mind was yesterday because the message Reformat command pulls up the subject to the first line, For example after reformatting a journal item it will appear like this:

1-OCT-75 2032-P JHB: Literature about SRI ARC given to HARPER'S

That is a typical view when using viespecs xb, Let me know how this works for you, Rita, 1

RH 23-OCT-75 13:17 26736

Message Reformatting

(J26736) 23-OCT-75 13:17;;;; Title: Author(s): Rita Hysmith/RH;
Distribution: /ESV([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: RH;

26736 Distribution
E. S. VonGehren,

RH 23-OCT-75 13:22 26737

The Dancing Girls' Return

Yes Paul, we are still around. We weren't quite sure if you all were still there or if the storm had leveled the place; however I saw old FGB last week and he assured me that you all we still "floating". By-the-way I finally ot that famous role of film developed. Do you think the guard would like some "reprints" of your place. Say hello to Dave, Anna and Phil. RH 1

The Dancing Girls' Return

(J26737) 23-OCT-75 13:22;;;; Title: Author(s): Rita Hysmith/RH;
Distribution: /PCB([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: RH;

26737 Distribution
Paul C. Bishop,

DAV 23-OCT-75 11:07 26738

More on the 'y' viewspec bug at ISI

Sandy, Here is some more information on the "y" viewspec bug. Garbaging occurs on the screen whenever the viewspec is on and I do any of the following. 1

Transpose two statements in the middle of the screen, regardless of the number of lines they contain. 1a

Append a statement containing two or more lines to one immediately above it. 1b

Copy/insert a statement, if the statement AFTER the place it is inserted contains "many" (?) lines. 1c

Delete text from a statement causing the statement to become at least one line shorter. 1d

Insert text in a statement causing the statement to become at least one line longer. (only happens sometimes) 1e

When the screen is split with a vertical edge, delete a statement/branch/group from the middle of the right-hand window. 1f

DAV 23-OCT-75 11:07 26738

More on the 'y' viewspec bug at ISI

(J26738) 23-OCT-75 11:07;;;; Title: Author(s): David C. Smith/DAV; Distribution: /FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: DAV;

26738 Distribution
Special Jhb Feedback,

JBP 23-OCT-75 15:35 26739

The best of NLS

In response to CHIs reqyest for thoughts on the most significant features of NLS (see -- 26703,) i submit the following list: 1

Structured Files 1a

Command Parser provided ? help 1b

View Specs 1c

especially for limiting the view 1c1

Links for cross referencing files 1d

the Journal as a shared data base 1e

the close coupling to programming facilities, eg Content Analyzers and User Programs 1f

the Procedure replace feature for developing the system. 1g

Statement Signatures for source code 1h

The Output Processor is NOT good but there must be something to fill this function. 1i

JBP 23-OCT-75 15:35 26739

The best of NLS

(J26739) 23-OCT-75 15:35;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /CHI([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

26739 Distribution
Charles H. Irby,

OLP 23-OCT-75 15:40 26740

message of good things

THE PREVIOUS STTATEMENTS AREE A BUNCH OF BUNK 1

text message 2

message of good things

(J26740) 23-OCT-75 15:40;;;; Title: Author(s): Opal L. Power/OLP; Distribution: /CDH([ACTION]) FJS([ACTION]) ; Sub-Collections: NIC; Clerk: OLP;

26740 Distribution
Chuck D. Hall, Flynn J. Stubblefield,

Gunter Report for week ending 10/24/75

The Week Ending 10/24/75 1

66-1 1a

Major Pete Lambert has continued to run the rewrite of 66-1. He is presently waiting for proofs from George Lithograph so that he can send them to Washington to get approval of the new type size. Pete feels like he doesn't want to rush ahead with things until DA in Washington sees the new type size. 1a1

PR 1b

I continued to format the PR manuals and to make the final changes. The one manual is finished except for the table of contents and the second one is near completion. I will finish it next week. Lynne Simms and I met with another man in Pr and chose a new document that input will begin on next week. There is no deadline on this document in that it was just released and a new version will not be published until July. There is also a possibility that Mr. McLeod (the man who was in charge of the first document) will put another one on line. 1b1

Trimus 1c

Again in a small corner of Gunter, several people had begun to work on another document without first consulting anyone. This is a 100 page report on a project that was a failure. It will be in the decimal format and input and edited by one person who was trained by Susan this week. We will first have to undo all the things that they entered in the wrong format. 1c1

Training 1d

Susan did some advance training with several people here and she also gave the first and second level courses to 5 new people. Two women came from PR and they will be working on the new document as well as updating the Base Tops document that was finished a month ago. Another woman works in Quality Control and does the final editing on all the documents that are released from Gunter. Sgt. Crabtree's secretary was alo trained as well as the woman who will be doing the Trimus report. 1d1

Hardware 1e

No changes on the hardware except that will all the new users, people are beginning to feel the crunch of not enough terminals. Hopefully, the pressure will be placed on the right people and more terminals will be made available. At present

there are 8 possible slots on the ELF 5 direct hookups and 3 dial ups. Two modems are still expected but have been for over a month. 1e1

POOH 23-OCT-75 15:58 26741

Gunter Report for week ending 10/24/75

(J26741) 23-OCT-75 15:58;;;; Title: Author(s): Ann Weinberg/POOH;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: POOH;

26741 Distribution

Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Israel A. Torres, Jan H. Kremers, Susan K. Ocken, Raphael Rom, David C. Smith, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Ann Weinberg, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White

DAV 23-OCT-75 17:38 26742

Bugs in NLS at ISI

Move Statement <bug> <bug> <option> <backspace> <backspace> <bug> <bug> 1

causes the error: "reference to undefined interpreter variable" 1a

Move Statement <bug> <bug> <option> ? <backspace> 2

causes the display to go away 2a

DAV 23-OCT-75 17:38 26742

Bugs in NLS at ISI

(J26742) 23-OCT-75 17:38;;;; Title: Author(s): David C. Smith/DAV; Distribution: /FEEDBACK([ACTION]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: DAV;

26742 Distribution
Special Jhb Feedback,

BEV 23-OCT-75 20:41 26743

Jeannie--This is our safe-retreat version, Somehow it didn't make it when I tried to journalize it the other day, Jim--Here's a copy for you to access, It has COM directives in it in various stages of completeness, so I don't recommend it except for DRAFT purposes, BEV

ABSTRACT

The oNLine System, Version 8 (NLS-8) developed at the Augmentation Research Center (ARC) was brought to prototype operation. Improvement in fundamental design continued, but new attention was given to applications and technology transfers. Strategies included an effort to involve more and more users outside ARC, establishment of experts in NLS within user organizations, a training program, an online query system to inform users about NLS, revision of the command language, and operation of the Network Information Center. New developments include a simple calculator subsystem, multi-host journal system, a variety of options to tailor the system to different working conditions, file access controls, and a control meta language to make user interface writing easier and more flexible. ARC developed a microprocessor-device, the Lineprocessor, to enhance inexpensive displays for use with two-dimensional display NLS and reduce communication cost. ARC made NLS available on a subscription basis through an information utility with its own computer.

CREDITS

The work from 10 May 1972 to 30 June 1974 involved the following ARC staff (some of whom have since left):

Andrews, Don I.
Auerbach, Marilyn F.
Bair, James H.
Bass, Walter L.
Beach, Mark
Beck, Jeanne
Bondurant, Rodney A.
Byrd, Kaye
Cooke, Judith
Dornbush, Charles F.
Duvall, William S.
Engelbart, Douglas C.
Evans, David
Feinler, Jake J.
Ferguson, William R.
Glenn, Joy
Guilbault, Carol
Hardeman, Beauregard A.
Hardy, Martin E.
Hopper, J. David
Irby, Charles H.
Jernigan, Mil E.
Johnson, Sandy
Kaye, Diane S.
Keeney, Marcia L.
Kelley, Kirk E.
Kudlick, Michael D.
Lane, Linda L.
Leavitt, Jeanne M.
Lee, Susan R.
Lehtman, Harvey G.
Lieberman, Robert N.
Limuti, Donald
Lister, Priscilla M.
Martin, Karolyn
Maynard, David
Meyer, N. Dean
Michael, Elizabeth K.
North, Jeanne B.
Norton, James C.

Page, Cynthia
Parsley, Bruce L.
Paxton, William H.
Peters, Jeffrey C.
Prather, Ralph
Ratliff, Jake
Rech, Paul
Row, Barbara E.
Vallee, Jacques F.
Van De Riet, Edwin K.
van Nouhuys, Dirk H.
Victor, Kenneth E.
Wallace, Donald C.
Watson, Richard W.
White, James E.

TABLE OF CONTENTS

INTRODUCTION 4

TIME COVERED 4a

This report covers Contract F30602-72-0313, which extended from March 1972 through June 1974. 4a1

SUMMARY OF WORK UNDER THIS CONTRACT 4b

In this period the central development at ARC, the Online System (NLS), was brought to prototype operation with outside groups. Passing this milestone led us to undertake clarification of our mid-range goals, to make changes in the organization of ARC, and to give more energetic attention to a wide range of users through development of technology transfer, of system features, and of services. We here report on ARC's goals in terms of the "Augmented Knowledge Workshop" -- a computer-based set of tools for people who need to manipulate knowledge in their work. ARC has been organized into a development branch, whose work is of primary concern in this report, and an applications branch, dedicated to offering NLS as an information utility; each is under an assistant director. 4b1

The International Conference on Computers in Communications took place in Washington in October of 1972. It was an important event for most of the research organizations associated with the ARPA Network; it was particularly important to ARC. The Network Information Center prepared informative directories of ARPANET participants, and published scenarios of many systems demonstrated at the conference. Half a dozen ARC staff members spent full or substantial part time preparing for NIC services at the conference, and for demonstrating ARC functions through the Network, and took part in a variety of other support functions. Twelve members of the staff were in Washington for the duration of the conference. 4b2

Development of NLS 4b3

NLS User Interface 4b3a

We have made the NLS user interface simpler, more

flexible, and easier to use. We completed design and implementation of a Command Meta Language and command interpreter system that allows creating commands in terms of what they do rather than in our programmers' language (high-level language command specification). The CML system compiles the high-level terms used to describe commands into a tree of instructions to drive the existing NLS command interpreter, centralizing both command parsing and feedback to the user. 4b3a1

This approach allows experiments with different command language structures and feedback, simplifies building subsystems, and allows users to tailor command languages for themselves.

It also allows NLS "frontend" functions to move to a minicomputer. During this contract period, before the move to the minicomputer took place, the new architecture resulted in more compact source code and more efficient running.

Other Changes 4b3a2

With the creation of the CML and our two years of experience with ARPANET users, we redesigned the command language to make it more consistent, and added features oriented toward novices.

NLS functions were reorganized into cleanly interconnecting subsystems. New subsystems include an arithmetic calculator integrated with NLS text files, the Modify subsystem, which contains automatic editing commands, and the Publish subsystem, which creates references, tables of contents, and the like.

We added a User Profile where a user can specify defaults such as the amount of feedback she gets, function of control characters, size of printout, type of recognition, and so on.

We added help commands that provide either a quick list of alternatives, complete command syntax, or access to complete, queriable documentation at a point related to what the user was doing when she asked. Cues to what the user was doing are derived from the CML.

User programs in the L10 programming language became

increasingly important as the world of NLS applications widened, and programs supported by ARC were integrated into documentation and organized into a directory.

restricted NLS Macro facility based on the command language but lacking in loops was implemented.

Provisions have been made to restrict access of NLS files to a list of idents selected by the file owner.

Dialog Support 4b3b

We integrated the NLS Journal into the ARPA Network Mail System both for input and output; we have taken a leading role in creating a Network Mail Protocol. 4b3b1

We designed a Distributed Journal System and associated network protocols that allow various Journal functions such as distributing, recording, cataloging, storage and retrieval to exist and cooperate on scattered hosts. 4b3b2

We implemented an initial system in which two Network-based PDP-10's cooperate in supporting a common Journal system. 4b3b3

Privacy provisions were added to the Journal. A user may restrict access to a list of idents she supplies. Private items are not cataloged. 4b3b4

Display Concepts and Terminals 4b3c

Display NLS was made more available to users working through the ARPANET or otherwise working remotely. In 1972 it became available through Imlac terminals; in 1973 we developed an inexpensive microcomputer based box, the Lineprocessor. 4b3c1

The Lineprocessor and associated software allow cheap, mass produced alpha-numeric terminals to display NLS files in the optimum, two-dimensional manner integrated with the Mouse and Keyset. The Lineprocessor does not require modification of the terminal.

This work included extension of the NLS virtual terminal concept and development of associated communication protocols for the ARPANET. The results

were useful in the development of the Network Graphics Protocol.

The Lineprocessors are being produced commercially for about $2000 per unit and were just coming into use on the Network at the close of the contract period.

Operating System 4b3d

The backup file archival and dump system, BSYS, developed at ARC, was released to the TENEX community. 4b3d1

A group allocation scheme to control logins to the system was built, put into operation, and released to the TENEX community. It split the users into groups and limits the number that may log in from each group. Allocation may vary during the day. Provision is made for brief "Express" logins over normal allocation. 4b3d2

Changes in TENEX necessary to support the Typewriter and Display versions of NLS became part of the standard BBN release of TENEX to allow future support of NLS on any standard TENEX. 4b3d3

We have found it advantageous to make several changes in our own TENEX, notably in the scheduler, while still remaining in harmony with BBN's standard TENEX releases. 4b3d4

We have built a system, Superwatch, to collect information on the consumption by various procedures and by users of CPU and clock time. 4b3d5

Network Information Center 4b3e

During the contract period, the Network Information Center was the main source of information about the personnel, computing facilities, and organizations associated with the ARPANET, and of a large volume of related data. It was also an innovative development in providing information to a community of computer users, in online, offline, and mixed form. Service included support and cataloging of online dialog (through the Journal), an online database and query language, dissemination in hard copy of a Resource Notebook, an ARPANET directory, and Network protocols, frequent tours for visitors, and response to questions from the computer public. 4b3e1

At the end of the report period, the operations of the Network Information Center were curtailed from the experimental array of NLS-based information exchange services to maintenance of directories of persons and resources for the ARPA network; a detailed account and evaluation of Network Information services is to be published as a separate technical report [1]. 4b3e2

THE ORGANIZATION OF THIS REPORT 4c

Since 1970, the central funding of ARC's work has been a series of ARPA contracts. The resulting series of reports (7101,) (5139,) and (13041,) [2] [3] [4] outlined the evolution of ARC and the development of NLS in those years. By the middle of this report period, however, a prototype Knowledge Workshop existed, and much of ARC's thinking, particularly planning, turned toward defining new goals and opening applications in different directions. The support of ARC in the fiscal year beginning in July 1974 is more widely spread than ever before, a trend that we expect to see continue. 4c1

One result of the evolution of project emphasis is reflected in the organization of this report. The work during this contract period is reported under four headings: 4c2

Chapter I: Application Experience 4c2a

Chapter II: User Interface 4c2b

Chapter III: NLS Subsystems 4c2c

Chapter IV: Workshop Foundation 4c2d

The detailed descriptions under these four headings are reported in a series of what amount to individual papers. Inevitably this approach, while preserving the work of individual researchers, leads to a certain amount of redundancy. We apologize; however, this format seemed appropriate for work that reflects a core of accomplishment, but is sufficiently diversified that write-ups aimed at specialized audiences are appropriate. 4c3

To take advantage of the automatic reference search of our

online system, bibliographic citations in this report look a little unusual. They will appear in two forms: 4c4

A string of numbers and letters in parentheses or angle brackets [e.g., <9an>] cites some other part of this report as identified by the statement numbers printed to the right of the page. Online, a reader may cite such an address and move automatically to the appropriate part of the report. 4c4a

A number in square brackets (e.g., [2]) cites a reference that is listed at the end of that particular section, in which bibliographic information about these documents is supplied in the usual way. Each reference in turn cites the statement where the reference has originally been cited. The four or five digit number at the end of the reference citation itself is the ARC catalog number. All of the documents cited in this report are either online or archived, and an online reader may move to that file automatically. 4c4b

A glossary of NLS-8 terminology and associated concepts has been published [5]. 4c5

REFERENCES 4d

[1] (4b3e2) Michael D. Kudlick, Network Information Center, Augmentation Research Center, Stanford Research Institute, Menlo Park, California 94025, June 1975, (25088,) 4d1

[2] (4C1) (4c4b) John B. Postel (UCLA-NMC), Official Initial Connection Protocol (Document No. 2), Network Information Center, Augmentation Research Center, Stanford Research Institute, Menlo Park, California 94025, 11-JUN-71, (7101,) 4d2

[3] (4C1) Douglas C. Englebart, and Staff of ARC, Computer-Augmented Management-System Research and Development of Augmentation Facility, Augmentation Research Center, Stanford Research Institute, Menlo Park, California 94025, APR-70, (5139,) 4d3

[4] (4C1) SRI-ARC, Online Team Environment / Network Information Center and Computer Augmented Team

Interaction. Augmentation Research Center, Stanford Research Institute, Menlo Park, California 94025, 6-MAR-73. (13041,) 4d4

[5] (4C5) SRI-ARC. NLS-8 Glossary. Augmentation Research Center, Stanford Research Institute, Menlo Park, California 94025, June 1975. (22132,) 4d5

Chapter I: APPLICATION EXPERIENCE

Aspects of ARC's Technology Transfer Strategy
(by Richard W. Watson, Douglas C. Engelbart, and James C. Norton)

5

INTRODUCTION

5a

By 1972, following the connection of the ARC computer system to the ARPANET and the establishment at ARC of the Network Information Center (NIC), we began to actively plan for and carry out an explicit technology transfer strategy [1]. Previous experience had indicated traditional approaches to technology transfer--publishing papers and reports, giving demonstrations at conferences and at SRI, making movies, and giving slide shows and talks--while useful, were not enough to achieve technology transfer at the rate desired. Additional mechanisms were needed, particularly, hands-on experience by target groups. This chapter outlines some of the additional mechanisms being used and considerations for their selection.

5a1

Discussion

5a2

At the heart of our views on technology transfer is the belief, based on experience, that the type of information system we are developing can only be developed and evolve in an environment with real users doing their everyday work on the system. We at ARC had been the prime users of the system over the first decade of its development, but in the last three years have begun to seriously enlist outside users from a variety of organizations. The importance of obtaining views and feedback from the users with a variety of needs from many organizational environments is vital to the ongoing healthy evolution of a flexible and general-purpose knowledge workshop. Based on this premise, we have taken four steps: 5a3

1) We organized our internal activities during this contract period into three areas that we call:

a) Analysis
b) Development
c) Applications

5a4

The functioning of these three parts as a harmonious whole

constitutes our research process. Development creates new user features, system organizations, and usage methodology based on experience and anticipated needs. Applications provides computer and other services, such as training to real users, both internally within the project and to outside groups. Analysis studies, at many levels, the ongoing system evolutionary process. 5a4a

2) We have set up an ARPANET-connected facility managed by Tymshare at their Cupertino, California, computer center to serve as a reliable utility for delivery of workshop computer services developed at ARC and elsewhere. The present PDP-10 system is called Office-1 and is accessible through the ARPANET and directly through low-speed or high-speed phone lines. As part of the delivery system we have also developed a low-cost unit called a Lineprocessor (now commercially available) to support the display version of NLS from low-cost commercially available alphanumeric CRTs [2]. 5a5

The ability to offer reliable computer service is crucial to the Development-Application-Analysis strategy. Staff and facilities with the know-how and motivation to create such a facility are not easily maintained by a highly Development- and Application-oriented organization such as ARC. Therefore, an important decision was made in 1972 to subcontract computer facility management to a corporation like Tymshare that has the staff and physical facilities for providing the needed services. Tymshare is responsible for hardware and operating system reliability. ARC is responsible for all services at higher levels. 5a5a

This has been a valuable and trend-setting move within the ARPA R&D computer community. 5a5b

3) We are asking each subscribing organization to provide what we are calling a "workshop architect," whose prime loyalty is to the using organization (preferably a person from the using organization, although we will provide a person for that role if necessary) to plan and conduct a staged evolution of the technology and training appropriate to his organization. 5a6

The importance to successful technology transfer of having a person within the target organization who is familiar with his organization's needs and the outside technology has been clearly demonstrated in the works of Allen [3][4][5][6][7]. Allen has called such a person a

gatekeeper, and has shown that most technology transfer occurs through such people, usually operating on an informal basis. We are trying to formalize and make explicit this role. 5a6a

On the ARC side we have created the roles of architect liaison -- whose function is to help define and shape the subscribing organization's basic level of service; and applications liaison -- who assists in developing those specific applications suitable to each client. Both are to be generally knowledgeable about ARC technology and outside user needs. It is across these overlapping liaison and workshop architect roles that we hope to achieve effective transfer, while being supported by other technical, analysis, and training personnel as well. 5a6b

4) The technology was originally developed on the assumption that it would be used as the everyday working environment of its users and that therefore the users would quickly be of the expert category. Experience has shown that a) it will probably be some time before this is the case, and b) even where it becomes the case, there is a critical transfer phase. Therefore, we have begun during the past year to pay much more attention to levels of documentation, usage scenarios, help, novice language features, etc., to provide a spectrum of functions from new to experienced users. 5a7

Our experience indicates that conscious attention to technology transfer by an R&D group affects: 5a8

A) Its organizational structure 5a8a

B) The types of skills and roles needed 5a8b

C) Its R&D strategy. 5a8c

LET US NOW LOOK AT THE FIVE TECHNOLOGY TRANSFER ISSUES THAT LEAD TO THESE STEPS: 5b

1) Need for demand pull versus technology push 5b1

We feel that successful transfer takes place only when a real need is met. Just to have a clever new toy is not sufficient for a technology to stick. It must meet a real

need at a cost appropriate to the user's value in order for transfer to be realized. 5b1a

This need to understand real needs in the outside world and to try to determine how well we are providing value leads to the creation of an Analysis function to study needs and analyze how well we are meeting them. We brought in an experienced operations research person with little interest in the technology as a thing in itself to provide this perspective.

Because of changes in funding levels and pressing needs for trained personnel within the Applications group, we have temporarily halted the Analysis function. Recruiting people with the appropriate interests, training, experience, and motivation for the important Analysis function is a difficult task. It is a highly interdisciplinary function and is not easily filled by the present orientations of academic computer science, operations research, or psychology departments. 5b1b

Usage by real users with work and applications to do other than build the NLS system is providing us with the feedback and contact with real needs that we feel are necessary to help us operate more on the need-pull side of the technology pull-push spectrum. 5b1c

2) One has to know where one is with respect to the two questions: 5b2

Is one trying to show something is feasible? OR

Is one trying to show something meets a need and should be continued?

We feel the former was accomplished and that we are in the latter area, thus requiring a shift in emphasis from technology-push to need-pull. 5b2a

3) The ease of technology transfer is proportional to the risk and cost to the user in terms of total system, organization, work habit, and training he has to undergo to adopt the new technology. Technology transfer has been described as more of a battle than just a matter of communicating an idea. Our experience confirms this view. 5b3

To meet this issue we are asking user organizations not to

try to adopt our technology on a broad scale, but to find a subgroup to try first, learn the advantages and problems, and then develop people trained in its use to take the next steps. 5b3a

4) Transfer of our type of technology is most successful by transfer of people. Studies at MIT of developments done at MIT and their transfer to industry found that on the order of 90% of the successful transplants were achieved by students or faculty going to work for the organization, obtaining an understanding of the organization's problems, and then bringing in the technology he was familiar with. 5b4

Industrial firms transfer many of their people periodically for just these reasons.

It is not easy to transfer people from SRI to outside groups, nor do we have enough people to do that. This problem, when coupled with the motivation of the gatekeeper concept, supports our establishment of the workshop architect role [3][4][5][6][7]. 5b4a

In the future when we have our experiment off the ground, we may try to transfer ARC people to user groups for six months to one year, and vice versa. For the past three months and for the coming months, we have stationed one ARC person in Washington, D.C., where a number of NLS user organizations are clustered, to provide an approximation of such a role. We have found this close contact useful and important to the transfer process.

We would like internally to move our people through the Development, Analysis, and Application areas to help them obtain several points of view, as our technology transfer effort matures. 5b4b

5) To transfer a system such as ours, and even many of its ideas, requires much more than publishing papers and reports. One needs a gut feeling that only a demonstration or, better yet, hands-on experience can give. This has led us to encourage visitors to ARC and to set up the NLS utility to provide service to real users. One problem we have faced is the task of finding suitable low-cost, commercially available display terminals for NLS use. Thus, most outside users to date have had to use the typewriter version, which has quite different user characteristics and feel from the display version they see in use at ARC. To make the display version

more widely available, we have developed a special microcomputer-based box for use with commercially available alphanumeric terminals that enables them to be used without modifications as true two-dimensional display NLS workstations [2]. 5b5

CONCLUSIONS 5c

Experience to date indicates that the elements of a technology transfer strategy have put us on the right track, although there is much yet to be learned about the process. It has shown us that technology transfer can be made an explicit, conscious process and that the efficiency and effectiveness of technology transfer can be improved as a result. 5c1

REFERENCES 5d

[1] (5A2) Douglas C. Engelbart, Richard W. Watson, James C. Norton. The Augmented Knowledge Workshop. In AFIPS Proceedings, Vol. 42, 1973 National Computer Conference, pp. 9-21, 1973. (14724,) 5d1

[2] (5A6) (5b5) Don I. Andrews. Lineprocessor: A Device for Amplification of Display Terminal Capabilities for Text Manipulation. In Proceedings of the National Computer Conference, 1974, p. 257-265. (20184,) 5d2

[3] (5A7a) (5b4a) Thomas Allen. Technology transfer to developing countries: The International Gatekeeper. In ASIS Proceedings, Vol. 7, The Information Conscious Society, 33rd Annual Meeting, 1970, p. 205-210. (13959,) 5d3

[4] (5A7a) (5b4a) Thomas Allen. Technology transfer to developing countries: The International Gatekeeper. Massachusetts Institute of Technology. Feb-71. (13859,) 5d4

[5] (5A7a) (5b4a) Thomas Allen. Roles in Technical Communication Networks. Massachusetts Institute of Technology. 1970. (13977,) 5d5

I Application Experience
5 Technology Transfer

[6] (5A7a) (5b4a) Thomas Allen, Performance of Information Channels in the Transfer of Technology, Massachusetts Institute of Technology, 1966, (15538,) 5d6

[7] (5A7a) (5b4a) Thomas Allen, Information Flow in Research and Development Laboratories, Massachusetts Institute of Technology, Mar-69, (15539,) 5d7

BEV 23-OCT-75 20:41 26743

(J26743) 23-OCT-75 20:41;;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /JML([INFO-ONLY]) JEW([INFO-ONLY]) ;
Sub-Collections: SRI-ARC; Clerk: BEV; Origin: <
ARCDOCUMENTATION, FINALCOM,NLS;15, >, 22-OCT-75 19:51 BEV ;;;;
####;

(J26732) 22-OCT-75 17:00;;;; Title: Author(s): James E. (Jim) White/JEW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JEW; Origin: < JWHITE, DPSPAPER,NLS;5, >, 22-OCT-75 16:56 JEW ;;;; ####;

26743 Distribution
Jeanne M. Leavitt, James E. (Jim) White,

DRAFT: NLS-9 Argument Conversions

NLS-9 ARGUMENT CONVERSION 1

INTRODUCTION 1a

The following brief note describes how arguments from the Command Language Interpreter (CLI) are currently being transformed within the middle-end for the NLS-9 tool. It is assumed the reader is familiar with "The Middle End: A protocol Interface" (MJOURNAL,26694,1:wh). 1a1

NLS ARGUMENT CONVERSION 1b

The following table describes how arguments are passed from the Command Language Interpreter (CLI) to the NLS Back End. Each argument is passed in PCPB36 format. This is converted to a standard L1011 format by the middle-end, which is in turn examined and perhaps modified by the NLS specific argument conversion routine. The following table gives the PCP format of the arguments, and the format of the arguments as generated by the argument conversion routine. If the argument is not one of the following the argument is converted from PCP format to a default L1011 representation, as defined by the middle-end. List elements which appear inside of square brackets [], are optional. 1b1

Command words, or #"..." : 1b2

PCP: 1b2a

LIST (INDEX %Command word number%, CHARSTR %Command Word%) 1b2a1

NLS: 1b2b

LIST (INDEX %Command word number%, LITERAL %Command Word%) 1b2b1

Command Word with zero value, or #"UNDEFINEDSTRING" where UNDEFINEDSTRING is any string which has the value 0 associated with it. 1b3

PCP: 1b3a

CHARSTR 1b3a1

NLS: 1b3b

STRING 1b3b1

DRAFT: NLS-9 Argument Conversions

CML user typein: LSEL, or SSEL with OPTION TYPEIN 1b4

PCP: 1b4a

LIST (INDEX %Entity type%, CHARSTR %user typein%) 1b4a1

NLS: 1b4b

LIST (INDEX %Entity type%, LITERAL %user typein%, INTEGER %window-id%) 1b4b1

CML address selection: SSEL, DSEL or LSEL with OPTION ADDRESS 1b5

PCP: 1b5a

LIST (INDEX %1 -> Address expression%, INDEX %Entity type%, CHARSTR %Address expression%[, CHARSTR %Address expression%], INTEGER %Windowid%) 1b5a1

NLS: 1b5b

LIST (INDEX %Entity type%, ENTITY %Selected Entity%, INTEGER %window-id%) 1b5b1

CML point selection: SSEL, DSEL, or perhaps an LSEL of an Entity type unknown to the front-end. 1b6

PCP: 1b6a

LIST(INDEX %2 -> point selection% , INDEX %Entity type%, *POINT, [*POINT]) 1b6a1

Where *POINT is shorthand for 1b6a2

LIST (INTEGER %window-id% , INTEGER %string-id% , INTEGER %line-segment-id% ,INTEGER %character count%) / CHARSTR %address expression% 1b6a2a

NLS: 1b6b

LIST (INDEX %Entity type%, ENTITY %Selected Entity%, INTEGER %window-id%) 1b6b1

VIEWSPECS 1b7

PCP: 1b7a

LIST (INDEX %48 -> viewspecs%, CHARSTR %viewspec string%, INTEGER %window-id%) 1b7a1

NLS: 1b7b

BLOCK %address of block containing updated viewspec words (2 words)% 1b7b1

LEVELADJUST 1b8

PCP: 1b8a

LIST (INDEX %49 -> leveladjust%, CHARSTR %leveladjust string%) 1b8a1

NLS: 1b8b

INTEGER %relative level count% 1b8b1

NLS ENTITIES 1c

The following Command words are defined as selectors in NLS. As such they are valid arguments to LSEL's, SSEL's, and DSEL's. They define the entity types recognized by the NLS Back-End. 1c1

"TEXT" = 1 SELECTOR = TEXT, 1c1a

"CHARACTER" = 2 SELECTOR = CHARACTER, 1c1b

"WORD" = 3 SELECTOR = WORD, 1c1c

"VISIBLE" = 4 SELECTOR = VISIBLE, 1c1d

"NEWFILENAME"=6 SELECTOR = TEXT , 1c1e

"OLDFILENAME" = 7 SELECTOR = TEXT , 1c1f

"INTEGER" = 8 SELECTOR = INTEGER, 1c1g

"NUMBER" = 9 SELECTOR = INTEGER, 1c1h

"PASSWORD" = 10 SELECTOR = PASSWORD, 1c1i

"INVISIBLE"= 11 SELECTOR = INVISIBLE, 1c1j

"FILENAME" = 12 SELECTOR = TEXT , 1c1k

"BRANCH" = 26 SELECTOR 1c1l

POINT = pbranch TYPEIN = TEXT ADDRESS = TEXT, 1c1l1

"GROUP" =27 SELECTOR 1c1m

POINT = pgroup TYPEIN = TEXT ADDRESS = TEXT, 1c1m1

"PLEX" = 28 SELECTOR 1c1n

POINT = pplex TYPEIN = TEXT ADDRESS = TEXT, 1c1n1

"STATEMENT" = 29 SELECTOR = STRING, 1c1o

"LINK" = 30 SELECTOR = TEXT %should be FILENAME%, 1c1p

"NAME" = 32 SELECTOR = WORD, 1c1q

The entity types which have SELECTOR = an uppercase identifier, for example SELECTOR = CHARACTER, are selectable by the front-end CLI. This means that if a command has an LSEL of type WORD for example and the user points to a word on the screen, the CLI will pick out the selected word and pass it on to the tool as if the user had typed the word in. The other entity types are meaningful only to the particular tool involved, in this case NLS. 1c2

L1011 LIST ITEM TYPES 1d

Each L1011 List element has a descriptor associated with it . One field of this descriptor, the ledubv field, contains a type code for the element. The following table gives the list element types used by the NLS Back End. For each type the actual value of the ledubv field is given in terms of built-in constants. The value returned by the construct ELEM#list#[i] is also given. 1d1

INTEGER 1d2

ledubv: uinteg 1d2a

ELEM #list#[i]: The value of the integer 1d2b

INDEX 1d3

ledubv: uindex 1d3a

ELEM #list#[i]: The value of the index 1d3b

BOOLEAN 1d4

ledubv: uboole 1d4a

ELEM #list#[i]: 1 or 0 (TRUE or FALSE) 1d4b

STRING 1d5

ledubv: ustrin 1d5a

ELEM #list#[i]: address of the string 1d5b

BITSTRING 1d6

ledubv: ubitst 1d6a

ELEM #list#[i]: address of a block. The first word of the block contains a bit count, n. The succeeding (n+35/36) words contain the bitstring, padded to the right word boundary with zero's. 1d6b

LITERAL 1d7

ledubv: utpsbl 1d7a

ELEM #list#[i]: address of a block. The first four words of which contain two text pointers which point to the beginning and end of an L1011 string, followed by the L1011 string. 1d7b

ENTITY 1d8

ledubv: utpblo 1d8a

ELEM #list#[i]: address of a block of four words which contain two text pointers which point to the beginning and end of an entity within an NLS file. 1d8b

BLOCK 1d9

ledubv: ublock 1d9a

ELEM #list#[i]: address of the block. 1d9b

NULL 1d10

ledubv: unull 1d10a

ELEM #list#[i]: 0 1d10b

LIST 1d11

ledubv: ulist 1d11a

ELEM #list#[i]: address of the list. 1d11b

FILES 1e

The file (nine,nlsacnv,nls,) contains the source code for the NLS argument conversion. The file (nine,madata,nls,) contains global data used by the argument conversion procedures. 1e1

PCPB36 1f

The protocols used by NLS nine make use of the PCPB36 encoding for passing arguments and results. PCPB36 provides a method of encoding typed data strucyures sequitially into 36 bit words. Each PCPB36 data element has a header which contains among others the fileds pcptype and pcplen. The pcptype filed gives the type of the element, the meaning of thepcplen field depends upon the data type. The following table gives the PCP types and describes their encoding. 1f1

1 (EMPTY) 1f2

PCPLEN: 0 1f2a

Successive words: none 1f2a1

2 (BOOLEAN) 1f3

PCPLEN: 1 or 0 (TRUE or FALSE) 1f3a

Successive words: none 1f3a1

3 (INDEX) 1f4

PCPLEN: i (Value of index: in the range [1 to 2**15-1]) 1f4a

Successive words: none 1f4a1

4 (INTEGER) 1f5

PCPLEN: 0 1f5a

Successive words: value (1 full word two's complement) 1f5a1

5 (BITSTR) 1f6

PCPLEN: n (bit count) 1f6a

Successive words: (n+35)/36 words containing bit string (padded to right word boundary with zero's) 1f6a1

6 (CHARSTR) 1f7

PCPLEN: n (character count) 1f7a

Successive words: (n+4)/56 words containing ASCII string (padded to right word boundary with zero's) 1f7a1

7 (LIST) 1f8

PCPLEN: n (element count) 1f8a

Successive words: n PCPB36 elements 1f8a1

DRAFT: NLS-9 Argument Conversions

(J26744) 23-OCT-75 22:02;;;; Title: Author(s): David S. Maynard/DSM;
Distribution: /NPG([INFO-ONLY]) RWW([INFO-ONLY]) ;
Sub-Collections: SRI-ARC NPG; Clerk: DSM;

26744 Distribution

Susan K. Ocken, Raphael Rom, Jan H. Kremers, David C. Smith, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Robert Louis Belleville, Elizabeth J. Feinler, Joseph L. Ehardt, Jonathan B. Postel, Kirk E. Kelley, Karolyn J. Martin, David S. Maynard, Kenneth E. (Ken) Victor, James E. (Jim) White, Elizabeth K. Michael, Don I. Andrews, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, Richard W. Watson,

Debugging User subsystems in NLS-9 1

Introduction 1a

A user subsystem is a grammar for the CLI which communicates with an existing instance of the NLS-9 Back-End. The CLI will provide a parsefuction which will provide a grammar state switch, without switching Back-End context. Another parsefunction will return the user to his previous grammar. This facility within the CLI will be used for those NLS Subsystems which do not easily fit into the model of a separate NSW tool. This note describes a facility for loading the Back-end components of such a subsystem into the NLS program buffer. This should allow programmers to begin debugging user subsystems without the pain and cost of constantly loading new NLS Back Ends. 1a1

Grammars: 1b

Until the parse functions described above are available each user subsystem designer will have to maintain his own copy of the "NLS Grammar". This should not be a problem because these grammars should for the most part be mutually exclusive. I suggest that the file <nine,nlsgram,> be maintained as the BASE subsystem grammar. Subsystem designers should maintain their own copies of the source and object file for the grammar for their particular subsystem. The grammar for each user subsystem should contain the "LOAD PROGRAM" Command. This command may be copied out of <nine,nlsgram,>. The following sequence should then allow you to debug your subsystem grammar and back end. 1b1

Connect to directory relnine. 1b1a

Rename (or copy) your grammar file to be exec.gram 1b1b

run messg 1b1c

give the command "LOAD PROGRAM (mydir,myxroutines,subsys,)" 1b1d

This will load the back-end support object file into the NLS programs buffer and update the dispatch tables in the middle-end. One can then give other commands in the grammar which call procedures within the back-end support object file. 1b1d1

Subsystems: 1c

To make a file containing the L10 code for a user subsystem

into a file which can be loaded into the program buffer one must do two things: 1c1

1) Include a declaration of a variable whose name is the same as the file name. This variable should contain a table specifing the external names for externally callable procedures. 1c1a

2) Compile the file to an object file called filename,subsys. Here filename is the name appearing on the FILE statement, and the name of the variable containing the procedure name table. 1c1b

EXAMPLE: 1c2

```
FILE xuo %(arcsubsys,x110,) to (maynard,xuo,subsys,)%                1c2a
%Package table - defines externally callable procedures%             1c2b
   (xuo)% _ (                                                        1c2b1
      $"XUOCONTCHAR", $xuocontchar,                                  1c2b1a
      $"XUOCURCON", $xuocurcon,                                      1c2b1b
      $"XUODSPLY", $xuodsply,                                        1c2b1c
      0,0);                                                          1c2b1d
%Other globals used by the subsystem%                                1c2c
   DECLARE STRING msg[200];                                          1c2c1
   DECLARE pt REF;                                                   1c2c2
% xroutines and support routines%                                   1c2d
   (xuocontchar) PROCEDURE                                           1c2d1
      (device REF %for which device%,                                1c2d1a
      cont REF, %which control%                                      1c2d1b
      char REF, %characters%                                         1c2d1c
      echo REF);                                                     1c2d1d
         #msg# _ STRING(ELEM#device#[1]) , SP,
         STRING(ELEM#cont#[1]);                                      1c2d1d1
```

```
            &pt _ ELEM#char#[2];                                    1c2d1d2
            *msg* _ *msg* , SP , SF(pt) SE(pt);                     1c2d1d3
            &pt _ ELEM#echo#[2];                                    1c2d1d4
            *msg* _ *msg* , SP , SF(pt) SE(pt);                     1c2d1d5
            dismes(1,$msg);                                         1c2d1d6
         RETURN;                                                    1c2d1e
      END,                                                          1c2d2
      (xuocurcon) PROCEDURE (length REF);                           1c2d3
         &pt _ ELEM#length#[2];                                     1c2d3a
            %as long as numbers are passed as text %                1c2d3a1
         *msg* _ "curcon " , SF(pt) SE(pt) ;                        1c2d3b
         dismes(1,$msg);                                            1c2d3c
         RETURN;                                                    1c2d3d
      END,                                                          1c2d4
      (xuodsply) PROCEDURE                                          1c2d5
         (type REF, %right margin or wraparound%                    1c2d5a
         margin REF %column for type%);                             1c2d5b
            &pt _ ELEM#margin#[2];                                  1c2d5b1
            *msg* _ "dsply ", STRING(ELEM#type#[1]) , SP ,
            SF(pt) SE(pt) ;                                         1c2d5b2
            dismes(1,$msg);                                         1c2d5b3
         RETURN;                                                    1c2d5c
      END,                                                          1c2d6
   FINISH of xuo                                                    1c2e
```

Debugging User subsystems in NLS-9

(J26745) 23-OCT-75 23:19;;;; Title: Author(s): David S. Maynard/DSM;
Distribution: /DAV([ACTION]) ROM([ACTION]) JAC3([ACTION])
EKM([ACTION]) NPG([INFO-ONLY]) RWW([INFO-ONLY]) ;
Sub-Collections: SRI-ARC NPG; Clerk: DSM;

26745 Distribution

David C. Smith, Raphael Rom, Jan A. Cornish, Elizabeth K. Michael, Susan K. Ocken, Raphael Rom, Jan H. Kremers, David C. Smith, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Robert Louis Belleville, Elizabeth J. Feinler, Joseph L. Ehardt, Jonathan B. Postel, Kirk E. Kelley, Karolyn J. Martin, David S. Maynard, Kenneth E. (Ken) Victor, James E. (Jim) White, Elizabeth K. Michael, Don I. Andrews, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, Richard W. Watson,

DAV 23-OCT-75 23:27 26746

Party talk

A pot-luck xmas party sounds great to me. Count me in. 1

DAV 23-OCT-75 23:27 26746

Party talk

(J26746) 23-OCT-75 23:27;;;; Title: Author(s): David C. Smith/DAV; Distribution: /DMB([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DAV;

26746 Distribution
Delorse M. Brooks,

JBP 24-OCT-75 13:46 26747

Demo Planning Check List

what did i leave out ?

Demo Planning Check List

We need certain information in planning for any demonstration of NLS. The more time we have for planning the better the demonstration will be. We can only do a reasonable job if we have some information about the following: 1

- Who will be present 1a
  - What is the nature and interest level of the audience 1a1
    - Best would be a list of attendees with the technical speciality and organizational affilation of each attendee indicated. 1a1a
  - How many will attend and in what sized groups 1a2
    - The maximum number of observers for a demonstration at one NLS work station is 5. 1a2a
- When is the demo 1b
  - dates and times 1b1
- Where will it be 1c
  - room, building, address, city, state 1c1
- What is the room(s) like 1d
  - Size (number of people it holds, physical dimensions), Furnishing (tables, chairs, carprts, windows) 1d1
- What audio visual equipment is available 1e
  - how many NLS work stations 1e1
  - slide projector and screen (size) 1e2
  - movie projector 1e3
  - video monitors slaved to NLS work stations 1e4
- Which computers will be available 1f
  - We need to distribute all files needed for the demo to at least two and hopefully three machines 1f1
- What Work Station equipment can we count on 1g
  - how many NLS work stations 1g1

When will equipment be installed and ready for testing 1g2

How much time is allotted to NLS-8, Graphics, NLS-9 1h

Are hard copy displays desired 1i

These are particularly important in the documentation and docoument production areas as it will be difficult to convey even a notion of the finished product with an on line demo. 1i1

Will we have a dress rehearsal with other contactors 1j

Who is coordinating the effort 1k

Demo Planning Check List

(J26747) 24-OCT-75 13:46;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /EKM([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) RLL([INFO-ONLY]) SGR([INFO-ONLY]) JMB([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

26747 Distribution
Elizabeth K. Michael, James C. Norton, Richard W. Watson, Robert N. Lieberman, Susan Gail Roetter, Jeanne M. Beck,

JHK 24-OCT-75 13:30 26748

USE OF THE TI ASR733 CASSET TERMINAL FOR DEX INPUT

This is a revised version of my earlier note and includes information on network restrictions.

USE OF THE TI ASR733 CASSET TERMINAL FOR DEX INPUT 1

This document describes the method of operation of the Texas Instruments ASR733 terminal and the CASSETTE utility program for use as a defferred execution input system. 1a

The operation of DEX is not covered, as this system is described completely in another document(...). 1b

A brief description of the terminal hardware features needed is also provided. 1c

TERMINAL HARDWARE 2

In order to be used as an input device for the CASSETTE utility program the terminal must provide facilities to allow the program to control the operation of the cassett drives remotely. 2a

To this end, the terminal to be used for cassette transmission must be equipped with the following options: 2b

Remote Device Control (RDC) 2b1

Full USASCII Character set 2b2

Automatic Search Control (This option is not necessary, but is potentially useful). 2b3

In order to check whether an existing terminal is so equipped, check the large sticker sometimes found inside the cover of the machine, or with the machine supplier. 2c

NETWORK RESTRICTIONS 3

If this terminal is used for casset input via the ARPANET the following conditions must be met: 3a

1) If used via TELNET be sure to change the escape character and set "transparent mode". 3a1

2) If the connection is made via a TIP be sure that the input buffer size is at least 95 characters. This restricion differs slightly from that for other casset terminals due to the fixed-length block format of the tapes prepared by the TI casset hardware. 3a2

OFF-LINE PREPERATION OF TAPES 4

The terminal operators manual, "SILENT 700 ELECTRONIC DATA

USE OF THE TI ASR733 CASSET TERMINAL FOR DEX INPUT

TERMINAL, MODEL 733 ASR/KSR OPERATING INSTRUCTIONS", (T.I. Manual no. 959227-9701, Revision C, or later) is included here by refrence. 4a

All tapes are prepared in CONTinuous format as per the instructions in section 5-1.1 of the operators manual. Terminate the tape by typing ^Z<cr> and then turning the record control switch off to force the last record to be written onto the tape. 4b

Tapes may also be edited and played back off-line using the machine by following the instructions in sections 5-1.2 and 5-1.3. 4c

Tapes prepared (accidently, of course), in line mode may be re-formated into CONTinuous mode by following the instructions in sections 5-1.5 or 5-1.7. This duplication operation is not advised, however, since it has been found to sometimes introduce extranious characters onto the tape. 4d

USING THE CASSETTE UTILITY TO READ TAPES INTO THE SYSTEM 5

To read tapes into the system perform the following operations on the terminal switch panal: 5a

1) Set KEYBOARD switch to LINE 5a1

2) Set PLAYBACK switch to LINE 5a2

3) Set RECORD switch to OFF 5a3

4) Set PRINTER switch to LINE 5a4

Place the casset to be read into either of the casset drives. (The left-hand drive is drive 1, the right-hand, drive 2). 5b

Login to TENEX in the normal manner and invoke the CASSETT Utility program. Operation of the program is illustrated by the following dialog. (Upper case text is typed by the system, lowercase by the user). 5c

```
@log kremers                                                    5c1
 JOB 23 ON TTY21 21-OCT-75 13:14                                5c2
 PREVIOUS LOGIN: 21-OCT-75 12:12                                5c3
 [KREMERS HAS ONE OTHER JOB]                                    5c4
 TENEX WILL GO DOWN WED 10-22-75 2200 TIL THU 10-23-75 0500     5c5
```

```
 FOR PREVENTIVE MAINTENANCE                                    5c6
@casset                                                        5c7
CASSETT TO SEQUENTIAL UTILITY                                  5c8
CASSETT RECORDER TYPE (TYPE "?" FOR HELP)ti                    5c9
COPY TO FILE: titfl1,txt [New file]                            5c10
INPUT FROM FILE: tty:                                          5c11
                                                               5c12
ENTER CASSETT DRIVE NO, (1 or 2)1                              5c13
TYPE SPACE WHEN READY                                          5c14
REWIND (Y OR N)yES                                             5c15
   (... the system is now reading the tape)                    5c15a
MORE FILES (Y OR N)yES                                         5c16
COPY TO FILE: titf12,txt [New file]                            5c17
INPUT FROM FILE: tty:                                          5c18
ENTER  CASSETT DRIVE NO, (1 or 2)1                             5c19
TYPE SPACE WHEN READY                                          5c20
TYPE SPACE WHEN READY                                          5c21
REWIND (Y OR N)nO                                              5c22
MORE FILES (Y OR N)nO                                          5c23
@logo                                                          5c24
```

The above illustraites reading multiple files, The procedure would be similar for a single file. 5d

USE OF THE TI ASR733 CASSET TERMINAL FOR DEX INPUT

(J26748) 24-OCT-75 13:30;;;; Title: Author(s): Jan H. Kremers/JHK;
Distribution: /RWW([ACTION]) HGL([ACTION]) JCN([ACTION])
EKM([ACTION]) JBP([ACTION]) RLB2([ACTION]) JMB([ACTION])
MEH([ACTION]) JCP([ACTION]) RMS2([INFO-ONLY]) ;
Sub-Collections: NIC; Clerk: JHK; Origin: < KREMERS,
TICASSET,NLS;4, >, 24-OCT-75 13:27 JHK ;;;;####;

26748 Distribution
Richard W. Watson, Harvey G. Lehtman, James C. Norton, Elizabeth K. Michael, Jonathan B. Postel, Robert Louis Belleville, Jeanne M. Beck, Martin E. Hardy, Jeffrey C. Peters, Robert M. Sheppard,

JBP 24-OCT-75 17:51 26749

Visit Planned by Dickerson & Morgan

RLL can you handle this ? also GAS2 may want to be involved as the SRI architect,

JBP 24-OCT-75 17:51 26749

Visit Planned by Dickerson & Morgan

DR, Loren Dickerson (University of Alabama) and MacPherson Morgan (SRI-Huntsville) will be visiting on 31 Oct 75 arriving at 10:00 am, i suggeste a 2hr visit but they may wish to stay longer, Dr, Dickerson is gathering data for a paper on methods for information exchange between faculty, administration, alumni, and students in a university setting, Dr, Dickerson referenced a phone conversation with DCE on 6-NOV-73 recorded as (20086,). 1

JBP 24-OCT-75 17:51 26749

Visit Planned by Dickerson & Morgan

(J26749) 24-OCT-75 17:51;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /RLL([ACTION]) GAS2([ACTION]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) JHB([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

26749 Distribution
Robert N. Lieberman, Glenn A. Sherwood, Douglas C. Engelbart, James C. Norton, James H. Bair,

FE Weekly Status Report 1

DIA 2

In Progress 2a

Debugging display package in stand-alone environment. 2a1

Coding missing portions of display package. 2a2

Loading CLI on 11 and debugging L1011 runtime+compiler. 2a3

Done 2b

Found and fixed bug in L1011 concerning addressing. 2b1

Found and fixed bug in L1011 concerning IF/branching. 2b2

CHI 3

Worked mostly on getting a CLI that would load into 11 for debugging. Quite a lot of trouble here but stand alone 11 system now loads without problems (including a test grammar). We got it loaded into 11 on Thursday but it did not run. Dia found l1011 compiler problem in runtime/tty stuff (?). I found some l1011 problems in cli code. We are ready to push on to debug it some more now. Need to get an ELF that we can use the same way as standalone, however. 3a

Sick on Wednesday 3b

Presented semenar on Tuesday re CLI implementation. 3c

Worked on CLI-implementation appendix to final report. 3d

Brought up new FE for MCA which treats selections of type INTEGER differently. 3e

ANDY 4

COMPLETED 4a

Modification of compactor to produce grammars loadable on PDP-11 4a1

Modification of CML compiler to output compactable grammars 4a2

Must produce distinction between kw's and selector kw's 4a2a

Must produce distinction between local and global variables in grammar 4a2b

Coded temporary loader to interface cli and grammars between 10 and 11 4a3

IN PROGRESS 4b

Modify CML compiler and compactor for call help and kwvar 4b1

Loading and debugging CLI on 11 4b2

DAV 5

IN PROGRESS 5a

Continuing to integrate HELP. Little work actually done, due to jury duty and writing a HELP "think piece". However, next week I'll stop thinking and start working. 5a1

The HELP CML is in and works. The xroutines need to be debugged. 5a2

DONE 5b

Read all the CML documents. RLL suggests that I don't rewrite the CML manual; rather spend the time improving CML. I've been thinking about some possibilities. 5b1

FE Weekly Status Report

(J26750) 24-OCT-75 16:57;;;; Title: Author(s): Don I, Andrews/DIA;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: DIA; Origin: < NSW-SOURCES, 24-OCT-STATUS.NLS;3, >,
24-OCT-75 16:32 CHI ;;;;####;

26750 Distribution

Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman, James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, Israel A. Torres, Jan H. Kremers, Susan K. Ocken, Raphael Rom, David C. Smith, Buddie J. Pine, Andy Poggio, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Ann Weinberg, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White

DVN 24-OCT-75 20:26 26751

Contact Report: Marge Lambie of Bonneville Power Authority, Interest in Front End, GE Proposal

(Bonneville) Contact report 1

(DATE) October 20th 1975 1a

(BY) Dirk van Nouhuys 1b

(ATTENDEES) 1c

Dr Marge Lambie of Bonneville Power Authority 1c1

(MEDIUM) PHONE 1d

(WHERE) My office 1e

(ACTION-ITEMS) 1f

Mailed Documents listed below 1f1

(DISTRIBUTION) ARC-LOG Docplan 1g

(REFERENCES)(journal,32515,)(journal,25216,) 1h

(DOCUMENTS) Hard copy mailed and received 1i

(GIVEN) Proposal to GE for a Text Processing System (25930), Automation in Technical Document Production [slightly revised from (journal,33412,), NSW Front End Specs (33403), and Charles Irby's Paper on the state of the CLI (isic,irby,cli-imp,). 1i1

(RECEIVED) Date and documents received 1i2

(REMARKS)Rob Lieberman's sending her notice of the AKW Seminar occasioned her call, but she was not interested in attending, she just want to check in particuarlly to learn about Front End developments. She is still expecting to put out an RFP, some time early next year. I took the occasion to describe the G E proposal. She was interested. It sounded as if the volumn of work envisioned for her RFP had grown a little which brings it very close to the needs GE was trying to satisfy. 1j

DVN 24-OCT-75 20:26 26751

Contact Report: Marge Lambie of Bonneville Power Authority, Interest in Front End, GE Proposal

(J26751) 24-OCT-75 20:26;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DOCPLAN([INFO-ONLY]) ARC-LOG([INFO-ONLY]) DLS([INFO-ONLY]) ; Sub-Collections: SRI-ARC DOCPLAN ARC-LOG; Clerk: DVN;

26751 Distribution

Joseph L. Ehardt, Raymond R. Panko, James H. Bair, David R. Brown, Glenn A. Sherwood, N. Dean Meyer, Kathey L. Mabrey, Norman R. Nielsen, Thomas L. Humphrey, Robert Louis Belleville, Elizabeth K. Michael, Richard W. Watson, James C. Norton, Robert N. Lieberman, Pat Whiting O'Keefe, Douglas C. Engelbart, Dirk H. Van Nouhuys, James C. Norton, Log Augmentation, Duane L. Stone,

kirknotes

Fri,, Oct. 24
good afternoon 1

It's pretty late and I'm pooped, so this will probably be short. Not much news anyway. Thanks for the Bev notes--and I appreciated your sympathy/empathy about the irritation of taking notes. Jim N. was sick all week, but I am definitely going to have to lodge a formal complaint with him. 1a

There really hasn't been anything definite at all about the documentation reshuffling. Just vague hints and assurances that it needs to be looked at. Dick asked me what I would suggest but I didn't really come up with anything very creative. It would be good to put our heads together about it--especially since the KWAC meeting, because I now see it from a little different perspective. I guess the only politically cool thing to do with CERTAIN PERSONALITIES(Y) will be to change responsibilities, but they don't discuss this with me fortunately. 1b

I worked on the Lexicon most of Wed Thurs and Fri. It's fun but slow going. However, I think what I've built is pretty solid. I added some more hellnotes for you. I shd be able to get to core early next week. I'm a little overwhelmed when I think of tackling core, and contemplate how all of this is supposed to work together. By the way, the ELUSIVE HELP BUG is driving me nuts, as is that stupid glitch of having to Compact the files. I talked a little more to Dave this week--I'm afraid maybe he's like several other programmers around here--he doesN7t get too turned on about fixing bugs. He feels like his top priority now is to get help running in NLS 9. We'll all have a pow-wow on Mon. Some of the things on his list to fix for 8,5 are rather obscure. 1c

By the way, Jim B. sent a copy of his proposed future plans for documentation. It's in <mjournal, 26698,> if you're interested. 1d

Your courses sound like maybe you did, in fact, make up their titles,,, If they require as much work as their titles require attention to comprehend, you must be keeping busy. I certainly can understand your dilemma about taking the practical route (programming) vs. the far more interesting and impratical (philosophy). I feel absolutely stuck in that loop, and am still hoping that there is a third choice. Very sad to see what our culture feels like paying people to do. 1e

Well I'm off to the mountains for a couple of days. See you Monday. Bev 1f

kirknotes

(J26752) 24-OCT-75 20:46;;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /KIRK([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk:
BEV;

26752 Distribution
Kirk E. Kelley,

Oct 20 weekly report for DAV

DONE 1

Finally finished my "think piece" on HELP. (I said I'd be finished by the weekend, didn't I? I just forgot to say which weekend.) It took a fair amount of time and thought. It's always hard to say whether these things are worth it or not, but I will say this: It has been very useful to me to write down in a coherent fashion the issues involved in HELP and information retrieval in the AKW. For the first time I feel like I'm ready to make a specific contribution to this place. 1a

IN PROGRESS 2

Integrating HELP into NLS 9. The CML is in and works. The back end needs debugging. (Dave Maynard's adding "Load File" should help a lot.) 2a

Also fixing the bugs in the older versions of HELP as BEV and KIRK find 'em. Two down, N+1 to go. 2b

Next week I stop thinking and start working. (Got my shovel and my bit bucket -- but I really hate machine language debugging!!) 2c

ASIDE 3

Elizabeth, You might consider creating a file for the weekly reports of people reporting to you, divided as follows, and into which people could directly add their reports. That might save you some work. 3a

Week of Oct 20 3a1

DAV 3a1a

DONE 3a1a1

... 3a1a1a

IN PROGRESS 3a1a2

... 3a1a2a

... 3a1b

Week of Oct 13 3a2

... 3a2a

DAV 25-OCT-75 18:03 26753

Oct 20 weekly report for DAV

(J26753) 25-OCT-75 18:03;;;; Title: Author(s): David C. Smith/DAV; Distribution: /EKM([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DAV;

26753 Distribution
Elizabeth K. Michael,

DAV 26-OCT-75 21:11 26754

My ideas on HELP

Here is a list of thoughts, ideas, comments, criticisms, opinions and bull-shit that I have had concerning the HELP system over the past two months. I would like to stimulate a dialogue on whether a general information retrieval system would be useful to us, what form it should take, and how HELP relates to it. Come on, gang, let's here it.

DAV 26-OCT-75 21:11 26754

My ideas on HELP

This note is a summary of my thoughts on the HELP system during the last two months. It is a compendium of ideas rather than a coherent model. The intent is to provide a concrete object around which interested people at ARC can formulate a specific plan for the future. 1

Index 2

I. Basic Premises 3

(1) HELP's primary goals are 3a

(a) to tell the user WHAT QUESTIONS TO ASK 3a1

(you can't retrieve information unless you know what information exists); and 3a1a

(b) to ANSWER SPECIFIC QUESTIONS from the user. 3a2

(2) UNIFORMITY IS GOOD; special cases are to be avoided. 3b

ALL NLS FILES should be meaningfully accessible using HELP (modulo protection considerations). 3b1

Corollary: data bases written explicitly for HELP should not be too different from ordinary NLS files. 3b1a

NLS SHOULD BE AVAILABLE to the sophisticated user. 3b2

Corollary: same as above. 3b2a

Additional corollary: HELP IS A SUPERSET OF NLS, not a small intersection with it. 3b2b

(3) HELP is not a peripheral issue to NLS because techniques for retrieving and managing information are important in an "augmented knowledge workshop". 3c

As Stuart Brand says, access is power. With the facilities ARC has implemented over the past 12 years, we could design a very powerful system in which 3c1

- information retrieval provides the ACCESS; 3c1a

- NLS presents the FORM. 3c1b

HELP is just a special case of a general information retrieval system. 3c2

3d

II. Main suggestions 4

(1) HELP should be a special case of a general information retrieval/management system. My view of "giving help" is that when a user asks for HELP: 4a

(a) An information retrieval process (IRP) is started. 4a1

(b) IRP links to the relevant tool HELP file. 4a2

This automatic data base selection is the only difference between HELP and information retrieval in general. 4a2a

(c) If the user was in the middle of a command (e.g. <CTRL>Q), the context is passed to IRP to enable it to begin as specifically as possible. Otherwise IRP simply waits for user queries. 4a3

The uer makes requests in the standard IRP query language (called Wyatt?). 4a3a

(2) The system must be able to scan data bases in a variety of ways: 4b

(a) "reading sequentially from the beginning as a tutorial;" 4b1

(b) "entering at any point and finding the answer to a specific question;" (This is the only one that is done adequately in the current system.) 4b2

(c) "browsing through the database, a combination of the above;" (above three from the 1974 final report) 4b3

(d) showing outline views and contextual views (ala Kirk's "Whole Universe Catalogue"), and the usual NLS views. 4b4

(3) Augment the query language. Transfer the control over information formatting out of the data bases into the query program. 4c

Possibility: make the query language and the language for writing user programs be THE SAME. After all, content analyzer filters and HELP queries are both information retrieval requests. 4c1

(4) Use as much of existing NLS as possible. Get rid of the HELP sequence generator. Make HELP/IRP an ordinary subsystem again. If the "sophisticated" user wants to use NLS commands, we may have to give him a quick tutorial. 4d

This will have the admirable effect of reducing the amount of code in HELP while simultaneously increasing its power. The amount of code will be reduced because the NLS Jump command, viewspecs and bugging will automatically become available. 4d1

The information retrieval process (IRP) will have a whole set of commands. But when using it for HELP, we can still provide the current naive interface -- just type "h" for help, or <CTRL> Q. 4d2

(5) Provide a uniform access to ALL NLS files, regardless of whether or not they were written by HELP data-base builders. Make HELP files look more like ordinary NLS files. Eliminate the special-purpose links. 4e

This will have to be done if HELP/IRP is to be used to interrogate other data bases, and if HELP data bases are to be interrogated with NLS. 4e1

Example: The system should be able to scan the (userguides, locator,) in the same way that any other data base is scanned. There is useful information in it. 4e2

Example: The NIC's goodies should be HELP-accessible. 4e3

Example: One data base that ought to be online and available to NLS users (via HELP) is an annotated bibliography. (The bibliography in RA3Y's OSIS proposal (33672,) is a good model.)

This would be extremely useful to paper writers and to those doing surveys. 4e4

(6) Create a "yellow pages" of online data bases. This will start with first concepts and tell the HELP user WHAT questions to ask. 4f

This will provide a powerful unifying force for the various data bases existing now. 4f1

It will also furnish the framework within which tool builders can (a) inform the community of their tools, (b) provide access to them, and (c) provide access to the associated HELP files. 4f2

(7) Evaluation criteria for these suggestions should be SYMMETRY, UNIFORMITY, CONSISTENCY, POWER. 4g

The viewpoint can and should be one of "What will augment the AKW?". (Maybe this is obvious.) 4g1

(8) From a HELP standpoint, features from the "NSW shopping list" (also in helpd) that would be nice to have are: 4h

(a) Comments - Text %between percent signs% that would be invisible to the user unless he had a certain viewspec on. Subsystems such as HELP and the Output Processor could look for comments and use the text inside in their processing. 4h1

This would have the plesant effect of making Output Processor directives invisible when doing Output Quickprint. 4h1a

(b) Back links - Two-way links vastly simplify the problem of maintaining large data bases. In addition, many thing become easy that are presently impossible, such as the ideal of "dynamic network documents" -- documents which contain people's comments, comments on comments, notes, cross-references, etc. 4h2

Example: It would be nice to have in documentation the names of people responsible for different parts of the system. When one of the people leaves ARC or is reassigned, EVERY reference to his name must be found and modified. This is a practical impossibility without some sort of backlinking facility, in which a name can point to every occurrence of it. 4h2a

(c) Indexes to the data bases - Searching a hashed index is more efficient than searching a data base, but we have to give some thought as to what should go into the index, whether it should be structured or linear, etc. 4h3

4i

III. Defects in the current system 5

The following are the negative aspects of the current system that I have most frequently observed and heard from others. 5a

(1) It is too time consuming. BEV reports that this is by far the most frequent complaint of HELP users. 5b

There is too much redundant printing -- backing up causes previous view to be printed again. This is especially serious in TNLS. 5b1

The "fix" of printing only the first few characters on backup is not a theoretical solution. 5b1a

At a paragraph at a time, it takes too long to get to relevant information. Most queries require wading through too many views. For a typical example, ask for help with the command Update File Rename. 5b2

This is a symptom that the query language and viewspecs are too restrictive. 5b2a

(2) HELP does not go far enough in telling the user WHAT information is available. In my opinion, this is the most serious defect of the current system. I would use HELP in spite of all its other failings if I could find out what I want to know. We need to back off; assemble ALL the concepts, people, files, indicies, etc, available; and then organize them in a way accessible to HELP. (The NIC seems to be doing a good job at organizing; perhaps we can use it as a model.) 5c

Example: There is no journal access from HELP. How can I find ALL the documents known to the system on any given topic? To access the journal indicies requires NLS. Even if HELP could access the indicies, the format of the files is completely different -- certainly a source of confusion. 5c1

Example: There is no logical hierarchy starting from first concepts "on down"; e.g. 5c2

SRI 5c2a

Life Sciences Division 5c2a1

Function 5c2a1a

If we had an organizing framework, we would then have to think about how far "up" to go. That is, should we include something about SRI? Menlo Park? California? USA? The world? The universe? Mind of God? We probably should stop at SRI, but something on the bay area might be nice -- e.g. restaurants (cf. Stanford AI's YUMYUM guide), entertainment, motels, etc. 5c2b

Since we would have this great information retrieval system, let's get some fun out of it. 5c2b1

If a system isn't stimulating, particularly an AKW system, there is something wrong with it. 5c2b2

(3) The HELP data bases are not in the usual format of NLS documents. Therefore they cannot be scanned (meaningfully) in the standard NLS ways. 5d

From the 1974 final report: "Note that portrayal of the

database in the Help command is controlled by the Help system and the database builders. In the rest of NLS, portrayal is controlled by the user with Viewspecs. In Help, embedded Viewspecs are inserted by the database builders." 5d1

If one looks at the core help file with NLS, he sees: 5d2

LEXICON: combination INDEX, GLOSSARY, and THESAURUS. 5d2a

A-colon: A: ADDRESS
##<address>## 5d2b

access: 5d2c

to files:
##<accessing>## 5d2d

to ARC:
##<entering>## 5d2e

accesslist:
##<Set !NLS !Private>## 5d2f

etc,, and later: 5d3

effects:
##<startup !effects>## 5d3a

Logout OK:
> The NLS command "Logout" causes you to leave both the NLS system and the TENEX Executive level at once. It is equivalent to using the Quit NLS command and then TENEX's Logout command. 5d3b

TNLS example:
BASE C: Logout
TERMINATED JOB #, USER,,, 5d3c

terminal commands
##<terminal-commands>## 5d3d

etc. The point is that this format makes sense ONLY when processed by HELP. It is not meaningful to the NLS viewer, the output processor, or any other subsystem. This is the most serious weakness in the way information is organized in the current system. It causes the following problems. 5d4

- NLS has had 12 years of experience designing effective

information-display techniques. All the work that has gone into those systems is discarded. 5d4a

- By building viewspecs into the data base, the builder PRESUPPOSES THE USES of the data base. The HELP user cannot look at documents in the way that is most convenient for him; he must look at them only in the ways provided by the builder. 5d4b

- HELP duplicates much existing NLS code (such as the sequence generator), making it a special-purpose piece of code, larger than necessary, with fewer features than are possible, hard to change (don't I know!), and an awkward fit with other systems such as the output processor. As NLS continues to evolve, HELP will remain a drag on our resources unless it is integrated more gracefully into the system. 5d4c

(4) Information display techniques are not well designed. 5e

There is no "scroll" command, i.e. "see next statement". (In fact, even DNLS has no facility for seeing the next window full of text if a statement is longer than one window.) 5e1

One ought to be able to go up and down in a document according to both logical and physical structure. 5e1a

There are no user-settable viewspecs, no ways of looking at the structure of documents. This is the biggest advantage NLS has over linear files; HELP disables it. 5e2

Kirk Kelley addressed this problem rather effectively with his "context views" in his "Whole Universe Catalogue". His work provides a good model here. 5e2a

5f

IV. What is HELP? 6

A user asks for help from an on-line system for one of two reasons: 6a

He wants information about the system, or one of the documents kept on the system. 6a1

He wants information about the state of his job. 6a2

HELP's task is very similar to NLS's: to organize information, provide the user with access to it, and present it to him in a

variety of representations. In this respect HELP ought to take advantage of the 12 years of development that have gone into NLS's display techniques. There is absolutely no reason why these techniques should be disabled. An additional requirement, however, is that HELP tell the user WHAT information is available. While NLS controls HOW information is presented, HELP controls WHAT information is presented. In fact, this is HELP's raison d'etre. If it can't do that, why have HELP at all? In accomplishing this task, HELP actually has two legitimate functions. I think of them, anthropomorphically, as (1) that of a LIBRARIAN, and (2) that of a CONSULTANT. The principal emphasis to date has been on the consultant aspect. But for a new user (such as myself) a system to tell what documents are available, and in general "what is going on," would be a tremendous, you guessesd it, help. 6b

A sub-function of the consultant is to act as a TUDOR, suggesting exercises and providing an environment in which the user can try them out. 6b1

Librarian 6c

The "librarian" tells the user what questions to ask. The user can then ask the "consultant" the questions, unless he is satisfied just knowing the categories of information. 6c1

Questions the "librarian" should be able to answer (the form of the query is dealt with in part V - the query language): 6c2

What documents are available? 6c2a

(will give the user a categorized list of EVERYTHING available) 6c2a1

Are there any documents on topic X? 6c2b

What is the latest document on X? 6c2c

Give me everything written on X (optionally in historical order). 6c2d

Show me the categories of information in the library. 6c2e

(i.e. how is the "world" classified?) 6c2e1

Who is working on/responsible for a given aspect of NLS? 6c2f

Who might be interested in topic X? 6c2g

My ideas on HELP

Who might give me help with X? 6c2h

<questions about NIC data> 6c2i

What is NIC? 6c2i1

What can I find out from it? 6c2i2

<questions about ARC, SRI, the Bay Area> 6c2j

What does ARC do? 6c2j1

What are the divisions of SRI? 6c2j2

Where is the action around here? 6c2j3

<questions answerable from other data bases: computer science bibliography, management information systems, ... whatever has been written by someone> 6c2k

Has anyone written a tool for translating English into Sanskrit? 6c2k1

What are the commands it will accept? 6c2k2

The "librarian" provides a logical framework for organizing information. (The physical framework is provided by NLS.) Many of the above questions can be answered already in one part or another of the system. The trick is to set up a UNIFORM ACCESS to them so that the information can be found without having to ask half a dozen different people and go through half a dozen different protocols. 6c3

Consultant 6d

The "consultant" answers specific questions when the user knows what to ask. 6d1

Questions the "consultant" should be able to answer (the form of the query is dealt with in part V - the query language): 6d2

What is the syntax of the INSERT command? 6d2a

What is the syntax of all NLS commands? 6d2b

(FOR ALL STATEMENT s IN base-subsystem-documentation DO 6d2b1

IF command(s) THEN show(syntax(s))) 6d2b1a

Show me examples of all NLS commands. 6d2c

(FOR ALL STATEMENT s IN base-subsystem-documentation DO 6d2c1

IF command(s) THEN show(example(s))) 6d2c1a

Tell me the function of all NLS commands. 6d2d

(FOR ALL STATEMENT s IN base-subsystem-documentation DO 6d2d1

IF command(s) THEN show(function(s))) 6d2d1a

What commands manipulate characters? Words? Statements? 6d2e

What commands manipulate statements but not words? 6d2f

What commands deal with windows? What is a window? 6d2g

What are the differences between TNLS and DNLS commands? 6d2h

Retrieve and show me the TECO manual. 6d2i

<specific questions about NIC data> 6d2j

What computers does SU-AI have? 6d2j1

<specific questions about ARC, SRI, the Bay Area> 6d2k

What does DAV do? 6d2k1

Who is in charge of user documentation at ARC? 6d2k2

What is a good Chinese restaurant in Palo Alto? 6d2k3

<specific questions answerable from other data bases: computer science bibliography, management information systems, ... whatever has been written by someone> 6d2l

What has been published on pattern matching in the last five years? 6d2l1

6e

V. The query language, or what's in a name? 7

Possibilities for the query language: 7a

English -- unrestricted natural language 7a1

Phrases (possibly whole structures?) 7a2

Boolean combinations of keywords 7a3

Keywords 7a4

Menu numbers 7a5

Formal language 7a6

Programming language 7a7

The first job we face in designing the user interface to such a system is to define the limits of our ambition. In its most general form the problem of giving help involves natural language understanding, epistimology, information retrieval, psychology, etc. The question is, what is the simplest thing we can do that will make an interesting contribution to the AKW?. 7b

(1) Natural language 7c

Beginning with Green et al's BASEBALL system, the idea of interfacing in natural language has stimulated much research. Today natural language understanding is an active area in AI (Woods, Colby, Winograd, Bobrow, Schank, Simmons); but a comprehensive solution is many years off, and the possibility exists that there may be no practical (= small, fast) solution. Therefore ARC should probably not open that can of worms. 7c1

If we did want a natural language interface, it should in the short- and medium-terms be a FORMAL LANGUAGE which looks like a subset of English. The most promising approach appears to be to use "production systems" or "rewrite rules". Several people including myself have developed production systems, and the technology is now well established. The big advantage of such systems is that rules can easily be included to handle special cases; e.g. 7c2

I WANT HELP WITH <PHRASE>:P --> <RETRIEVE :P> 7c2a

WHAT DO YOU KNOW ABOUT <PHRASE>:P --> <RETRIEVE :P> 7c2b

WHO IS WORKING ON <PROJECT>:PRJ --> <RETRIEVE <WORKERS :PRJ>> 7c2c

In this way a RELEVANT SUBSET of a natural language can be assembled -- just those forms which we experimentally decide are necessary. 7c3

If we were to go this route, perhaps we could also use this production language as the new CML (if there is one). In that case things would really start to get unified. The production language would be: 7c4

(a) the language in which tool builders write communication protocols with the terminal (the CML); 7c4a

INSERT <TEXT-ENTITY>:T <ECHO "TO FOLLOW"> <DESTINATION>:D 7c4a1

--> <XINSERT :T :D> 7c4a1a

(b) the language in which tool builders specify HELP protocols (the query language); 7c4b

WHAT DO YOU KNOW ABOUT <PHRASE>:P ? 7c4b1

--> <RETRIEVE :P> 7c4b1a

(c) the language in which users write programs (the user programming language). 7c4c

(2) Formal languages 7d

perhaps the query language should not be English or even English-like. Edmundson and Epstein suggest using a programming language for a retrieval language, with English words as data objects. For example, 7d1

FIND insert AND statement 7d1a

finds the first statement containing both the words "insert" and "statement". The command 7d2

FIND ALL insert AND statement 7d2a

would act like a content analyzer filter, filling the display screen with statements meeting the specified criteria. 7d3

This is another promising "user language" for writing programs. Again, making it the HELP retrieval language produces one uniform formalism for both. 7d3a

This retrieval system can be used both by the user to interrogate the data base and by the builder to verify its capabilities. Since it is a programming language, either one can write a program to interrogate the data base in sophisticated ways or to verify its completeness or consistency. 7d4

Programming languages provide the user/builder with a great deal of power. But since he seldom wants to do anything complicated, simple querys must (and CAN, if the language is well designed) be kept simple in form. 7d5

(3) NLS considerations 7e

With NLS a large, well-structured data base already exists, with well-defined accessing and processing primitives (the NLS core routines). So our information management problem is considerably simpler than in the general case. The facilities needed to make NLS's primitives easily available to the general user are: 7e1

- a more flexible user interface, 7e1a

- a more uniform access method, and 7e1b

- some automatic management system for maintaining consistency, completeness and currency. 7e1c

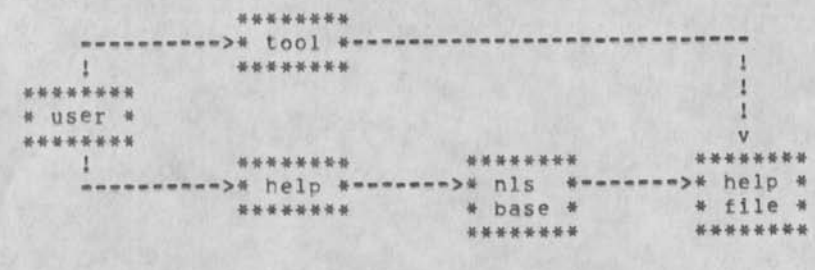
7f

VI. Retrieval 8

The issue of what HAPPENS when the user makes a query can, at least conceptually, be separated from the issue of the FORM of the query. (I'm not sure this can or should be done in the actual implementation.) This section deals with the retrieval half. 8a

(1) Communication with NLS could be accomplished by the following organization. 8b

Invoking HELP sets up a HELP process, which sets up an NLS sub-process to do the actual processing of the HELP file. 8b1



8b2

Perhaps this could be implemented using two new viewspecs, WHAT?? MORE VIEWSPECS?? (This may be a bad idea.) 8b3

Viewspec "M" 8b3a

Put menu numbers (small integers) instead of statement numbers (3A2B5C2) by statements. If we did this, I would propose that menu numbers monotonically increase from the time the viewspec goes on, so that (especially for TNLS) old views would not have to be regenerated to refer to a menu item. 8b3a1

Viewspec "st" (s followed immediately by t) 8b3b

Show all lines in current statement, followed by one line from each sub-statement (one level down only). In other words, build the menu-generating capability into NLS. This simplifies things like bugging, jumping and printing. 8b3b1

The HELP process interprets everything typed by the user, turning user-format requests into NLS commands. It provides several "escape" facilities: 8b4

@ = execute one command in NLS (could be Goto Base or other subsystem). This is intended to make NLS available to sophisticated users. 8b4a

= execute one command in the tool for which we are currently giving help. This is the "tudor" function. 8b4b

If the tool is NLS itself, then # is the same as @. 8b4b1

This could also be accomplished by using the @ escape (so that only one escape is really necessary), and then using the NLS Execute command. However, this requires the HELP user to know NLS. 8b4b2

(2) It certainly seems that the HELP data base builder ought to be able to add and modify information using the SAME system which retrieves it, so that he always sees what the user sees. 8c

Yang suggests an information storage/retrieval/updating system based on eight commands: STORE, RETRIEVE, ADD, DELETE, REPLACE, PRINT, COMPRESS, LIST. 8c1

(3) To improve the efficiency of STORE/RETRIEVE, we could use a hash table. To keep the standard NLS file structure intact, HELP files could have an "index" -- a separate file containing the hash

links. The index would be maintained automatically by system routines. (Several people have previously suggested this.) 8d

Yang shows that a linked-hash-table algorithm like LISP's oblist requires an average of 1.5 searches regardless of table size, versus 15.6 for a binary search of a 50K-item table or N/2 for a linear search. 8d1

(4) NLS availability 8e

The NLS jump and display (viewspec) commands definitely should be available to the HELP user. This suggests that HELP should go back to being an ordinary subsystem. I think it should. It would increase the compatability of HELP with the rest of NLS, and follows the AKW philosophy of making things uniform. 8e1

The information retrieval aspects certainly seem subsystem-like. 8e1a

We could provide a smooth entry into the system when the help aspect is used, such as automatically loading the subsystem and interpreting his typein. The naive HELP user would never know that he was talking to a subsystem. 8e1b

NLS is tailored to display useful views of information. HELP might take advantage of them as follows. Again the naive user need never know that NLS was doing the work for him; but the sophisticated user would be able to take advantage of it. 8e1c

View Name	NLS equivalent	
OUTLINE	x(s) viewspec	8e1c2
CONTEXT	Jump Up x(s)	8e1c3
DETAIL	w viewspec	8e1c4
MENU	"st" viewspec	8e1c5

While the HELP data bases should be meaningful NLS files, perhaps we could incorporate into the text (e.g. with a "comment" construction) data or program that would augment the text when it is accessed with the query system. This could be used to generate a more relevant view for each particular user. 8e2

(5) Possible implementations 8f

Current -- We could leave things as they are, except for fixing bugs and minor additions. 8f1

This has the advantage of requiring the least amount of work. 8f1a

Woolley's (see the appendix) -- We could structure the data bases into a network of relationships and interrogate them with boolean combinations of the relations. 8f2

There are some good ideas here. 8f2a

There are many systems that do this kind of thing, but they are all pretty artificial. 8f2b

ZOG -- We could put procedures into the text which could be activated by the query system. 8f3

This is a powerful idea in any system. 8f3a

We could implement it with %commented text%. 8f3b

- Not visible unless a certain viewspec is turned on. 8f3b1
- Can contain any number of good things. 8f3b2

Back links 8f3b2a

Code for processing the text 8f3b2b

Output processor commands 8f3b2c

Debugger information 8f3b2d

Production system -- We could unify the notions of command specification language, query specification language, and user programming language into one language. 8f4

This is the biggest divergence from the current systems, but it also has the most potential for improvement: flexibility, versatility, power. 8f4a

Relevant maxims: Progress is a step function, not a continuous curve. "Throwaway software" is a good idea. 8f4a1

I favor this approach. 8f4b

8g

VII, Structure of the HELP data bases 9

Whatever the retrieval mechanism, we need a good metaphor for the organization of the data bases. I don't have a good model of them, and I doubt if most tool builders will. This must be resolved before we can hope to have consistency across all the different data bases. 9a

Possible models 9b

(1) Encyclopaedia 9b1

In this approach, descriptions are organized into short, self-contained "articles". As in any encyclopaedia, the articles need not be written by a single author, or even a small number of authors; "experts" on any topic can contribute articles. At the end of each article would be a set of cross-reference links. 9b1a

Advantages 9b1b

An encyclopaedia provides dynamic yet organized growth. As new concepts are developed, they can be integrated smoothly into the existing documentation. 9b1b1

Encyclopaedias stimulate browsing. 9b1b2

They can answer specific questions, while at the same time providing substantial context. Individual articles can be hierarchically organized. 9b1b3

Disadvantages 9b1c

The collection of articles does not have the sequential coherence of a novel. 9b1c1

An encyclopaedia cannot be hierarchically organized, and therefore it is not amenable to outline views. We would still have to provide a hierarchical index. 9b1c2

(2) Textbook 9b2

In this approach, documents would be organized like ordinary, preferably college-level textbooks. Each textbook would be self-contained, begin with first concepts, and progress through more and more advanced material. 9b2a

Advantages 9b2b

This is the way people have learned subjects all their lives. 9b2b1

Textbooks can be read front to back, as well as delving into the middle. 9b2b2

With NLS, a good hierarchical organization can be achieved. 9b2b3

Disadvantages 9b2c

It may be difficult to answer specific questions concisely. 9b2c1

Jim Bair recently recommended that a textbook be developed to teach NLS. If this is done, then 9b2d

(a) it should be integrated into the HELP documentation, since it would certainly be of use to HELP users; and 9b2d1

(b) it should be done online, since we are in the business of online systems. If it can't successfully be done online, then NLS needs changing. 9b2d2

(3) Thesaurus 9b3

A thesaurus is both a cosmology and a set of links to similar concepts. Explicitly presented in the front of a thesaurus is a complete hierarchical organization of all the concepts in nature. The terms in a language are then classified according to this hierarchy. 9b3a

A thesaurus is just an index, but we could have text rather than simply words in the body of it. An alternative might be to have the text organized into an encyclopaedia, with the thesaurus providing a structured access to it. I think this has a lot of promise. 9b3b

Advantages 9b3c

The thesaurus provides alternative ways of categorizing information. Frequently I want to view information in different organizations than were provided by the data base builder. 9b3c1

For example, I may in one instance want to know what NLS procedures display text on the screen, and later what procedures need to be changed if another viewspec is added. Some of the same procedures may be in each

set. In the first case I am probing the data base as a system user, and in the second as a system builder. 9b3c1a

The thesaurus approach breaks down concepts into "primitive" units. These primitive concepts can be combined by the user to formulate his retrieval request. The thesaurus provides a link organization that could be automatically processed by routines to yield a powerful question answering capability. 9b3c2

***** A very promising application is user-tailorable manuals. By extracting the text associated with a set of concepts, the user can assemble a coherent document containing only what he wants. This kind of versatility is going to be necessary to keep the sheer quantity of future documentation from getting out of hand. 9b3c2a

The concepts used in organizing the data base would be explicitly stated. Each tool builder would have to create a hierarchy of the concepts with which his tool deals. 9b3c3

This might provide the framework for an "automatic broker". The "broker" would automatically scan the concept hierarchy of each tool in searching for tools applicable to the user's problem. 9b3c3a

Disadvantages 9b3d

Thesauruses cannot be read front to back, unless the text is in a separate area. 9b3d1

(4) Dictionary 9b4

A dictionary would be a linear, alphabetized collection of terms with certain standard entries for each term. The equivalents of "noun", "verb", etc. might be "command", "text entity", "subsystem X", etc. This is fairly close to the current structure of the data bases, except that the statements associated with each term are not explicitly categorized. Also the notion of different "senses" for terms is not explicitly developed. 9b4a

Advantages 9b4b

Dictionaries are excellent for answering specific questions. 9b4b1

Disadvantages 9b4c

This organization is neither linearly readable, nor hierarchically displayable, thus limiting its versatility. 9b4c1

There is not explicit cosmology. 9b4c2

(5) Telephone directory (yellow pages) 9b5

Under a given concept (e.g. "moving text") would be a list of the applicable commands and/or references. This would probably be used as an index, with the text residing elsewhere (e.g. in an encyclopaedia); otherwise the problem of multiple references to a piece of text might become unmanageable. 9b5a

Advantages 9b5b

It would provide a conceptual classification for the concepts involved in a tool. 9b5b1

The concepts could be classified hierarchically, though yellow pages usually do not do this more than one level. 9b5b2

Disadvantages 9b5c

The format is pretty artificial for HELP data bases, although it might be useful for the NIC. 9b5c1

There is not enough redundancy in NLS to make for a very long list of alternative ways of doing things. 9b5c2

It cannot be read from front to back. 9b5c3

(6) Network 9b6

This is what we have now, a set of text fragments connected by links. 9b6a

Advantages 9b6b

Text fragments need occur only once; subsequent references can link back to the original and be automatically retrieved. 9b6b1

References across files are no more difficult than references within a file. 9b6b2

Disadvantages 9b6c

The data bases can be accessed in only one way: through the HELP system. In particular, they cannot be read from front to back. Browsing is difficult. More flexible access methods would be more useful. 9b6c1

There is no explicit (user examinable) hierarchy of concepts, though there could be with this organization. 9b6c2

Consistency and completeness 9c

As Harry Josselson writes, we want to organize each data base "to insure that words will be described (coded) with consistency, that is, to insure that questions which have been asked about certain words will be asked for all words in the same class, regardless of the fact that they may be more difficult to answer for some than for others." Some requests might be a simple lookup. Others, such as "FIND ALL ..." might require extensive processing, just as NLS can have simple viewspecs or complicated content analyzer programs. But all requests that can be handled for one term must be handled for all "similar" terms. (The idea of CLASSES of terms suggests itself.) 9c1

All requests should be made in the same language. 9c1a

In order to insure consistency, the tool builder should include a short list of the relations that hold between objects in his system and the text that describes those objects. (See Woolley's system in the appendix below.) 9c2

For NLS commands the list might be: SYNTAX, FUNCTION, EXAMPLE, TEXT ENTITIES MODIFIED, SUB-PARTS, SIMILAR COMMANDS, SEE ALSO. Each command would have a piece of text associated with every relation. The entry for the Logout command now becomes: 9c2a

Logout 9c2a1

SYNTAX 9c2a1a

Logout OK 9c2a1a1

FUNCTION 9c2a1b

The NLS command "Logout" causes you to leave both the NLS system and the TENEX Executive level at

once, It is equivalent to using the Quit NLS command and then TENEX's Logout command. 9c2a1b1

EXAMPLE 9c2a1c

BASE C: Logout
TERMINATED JOB #, USER,,, 9c2a1c1

TEXT ENTITIES MODIFIED 9c2a1d

none 9c2a1d1

SUB-PARTS 9c2a1e

none 9c2a1e1

SIMILAR COMMANDS 9c2a1f

<NLS, Quit> 9c2a1f1

SEE ALSO 9c2a1g

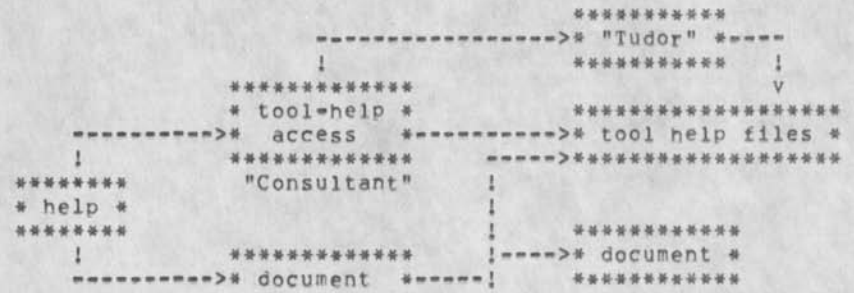
<NLS, Quit> 9c2a1g1

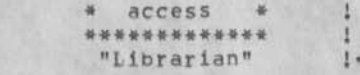
<TENEX, Logout> 9c2a1g2

The STORE function would prompt the tool builder for each element when he writes the HELP file, The "librarian" aspect of HELP would be able to supply the user with the list for any tool. 9c3

The access paths implemented will both determine and be determined by the file structure chosen, The overall organization will be something like: 9d

9d1





9d2

9e

VIII. Appendix - Example of an existing system with interesting features 10

The following is an information management system with some interesting features. Food for thought. It was designed seven years ago by G. H. Woolley of what was then SDS. It is described in a paper titled "Automatic Text Generation" in the proceedings of the 1969 International Conference on Computational Linguistics. The following are excerpts from that paper. 10a

TEXT SPECIFICATION 10b

A TEXT SPECIFICATION is a compact description of the outline of a text. The form of a text specification is as follows [I have put it into L10 syntax]: 10b1

<text specification> ::= 10b1a

<subspecification> $(; <subspecification>) 10b1a1

<subspecification> ::= <object name> '(<R> $(', <R>) ') 10b1b

<R> ::= <relation name> $1('(<R> $(', <R>) ')) 10b1c

<object name< ::= one or more contiguous characters 10b1d

<relation name> ::= 10b1e

one to three contiguous alphabetic characters 10b1e1

Object names are key words or phrases. They represent OBJECTS of interest within a data base, for example, names of commands in a programming language, people on a project or pieces of equipment in a system configuration. 10b2

A RELATION is a connection or association between one object and a FRAGMENT OF TEXT (i.e. a part of a sentence or one or more closely related sentences) and zero or more other objects. The following are typical relation names: NT (narrower term); PT (part); FUN (function); SYN (syntax); EG (example). 10b3

FIRST EXAMPLE 10b4

Consider the following request: 10b4a

Please create a text that explains the function and the syntax of the narrower terms of command. Examples of each command should be included. Each command should be discussed separately -- function, first, then syntax, and last an example. 10b4a1

This request can be stated briefly by the following text specification: 10b4b

COMMAND (NT (FUN, SYN, EG)) 10b4b1

The corresponding text that would be generated would have an outline in the following form: 10b4c

First Command 10b4c1

Function of First Command 10b4c1a

Syntax of First Command 10b4c1b

Example of First Command 10b4c1c

Second Command 10b4c2

Function of Second Command 10b4c2a

Syntax of Second Command 10b4c2b

Example of Second Command 10b4c2c

etc. 10b4c3

The output for one command in the outline might be: 10b4d

The function of the Set Command is to set a specified control parameter to a specified integer value. The format of the Set Command is: 10b4d1

<name> = <integer> 10b4d1a

An example of the Set Command is: 10b4d2

SL:OU = 100 10b4d2a

In the example, the maximum number of on-line users (SL:OU) is set to 100. 10b4d3

SECOND EXAMPLE 10b5

Suppose that in addition to the text of the first example, an introduction is desired in which a list of all the commands is given. The appropriate text specification would be: 10b5a

COMMAND (NT); COMMAND (NT (FUN, SYN, EG)) 10b5a1

THIRD EXAMPLE 10b6

Suppose that instead of grouping information by command, it is desired that all the functions should be grouped together, etc. Then, the appropriate text specification would be: 10b6a

COMMAND (NT (FUN), NT (SYN), NT (EG)) 10b6a1

DATA BASE 10c

A DATA BASE for a particular subject consists of two parts: 10c1

(a) a thesaurus that relates objects to each other and to fragments of text, and 10c1a

(b) fragments of text. 10c1b

THESAURUS 10c2

A thesaurus contains an entry for each object. 10c2a

An entry for a single object consists of any number of relationships. Each relationship relates the object to a fragment of text and, in some cases, to one or more other objects in addition. 10c2b

An object that is being focused on (i.e. as the object in a text specification or as the object that an entry is for) is referred to as a SUBJECT. 10c2c

An object should be included under a relation in a particular entry if it occurs in the fragment for that relation and its meaning is not self-evident in that context. 10c2d

The following is an example of an entry in a thesaurus for SET COMMAND: 10c2e

SET COMMAND 10c2e1

FUN: 10 10c2e1a

CONTROL PARAMETER 10c2e1a1

SYN: 11 10c2e1b

NAME 10c2e1b1

'=' 10c2e1b2

INTEGER 10c2e1b3

EG: 12 10c2e1c

FUN, SYN and EG are relations. The function of the SET COMMAND is stated in fragment 10, the syntax in fragment 11 and an example of the SET COMMAND is given in fragment 12. The object CONTROL PARAMETER occurs in fragment 10 and its significance is not self-evident in that context. 10c2f

TEXT FRAGMENTS 10c3

The data base includes a text fragment for each relationship in each entry. These fragments can be arranged (in the data base) into one or more unified texts, perhaps with some fragments left over. 10c3a

Fragments 10, 11 and 12 referred to above might read as follows: 10c3b

10: The function of the Set Command is to set a specified control parameter to a specified integer value. 10c3b1

11: The format of the Set Command is: 10c3b2

<name> = <integer> 10c3b2a

12: An example of the Set Command is 10c3b3

SL:OU = 100 10c3b3a

In the example, the maximum number of on-line users (SL:OU) is set to 100. 10c3b4

(Later in his paper Woolley discusses multi-file data bases.) Although each specification must be directed at a particular data base, not all (or even any) of the fragments in the resulting text would necessarily be from that data base. 10c3c

[Still later he discusses on-line documentation systems,] 10d

FIRST EXAMPLE 10d1

A new person has been assigned to an implementation project. He would like up-to-date documentation of parts of the system relevant to the work he will be doing. In different areas he wants different types of information. The structure of the texts generated for him can be tailored to his needs by use of appropriate text specifications. If he needs more information in some areas, he can use the system interactively. [I really identified with this example!!] 10d1a

SECOND EXAMPLE 10d2

The information in a particular area changes frequently and a number of people need to receive up to date information periodically. A text specification can be created to generate the appropriate information, and (assuming the structure of the data base doesn't change significantly) the same specification can be used to generate a text with the same structure (but different information) as often as is desired. 10d2a

10e

DAV 26-OCT-75 21:11 26754

My ideas on HELP

(J26754) 26-OCT-75 21:11;;;; Title: Author(s): David C. Smith/DAV;
Distribution: /HELP([ACTION]) SRI-ARC([INFO-ONLY]) ;
Sub-Collections: SRI-ARC HELP; Clerk: DAV;

26754 Distribution

James C. Norton, Jeffrey C. Peters, Dirk H. Van Nouhuys, Kenneth E. (Ken) Victor, Richard W. Watson, Don I. Andrews, David L. Retz, Laura J. Metzger, Karolyn J. Martin, Jan A. Cornish, Larry L. Garlick, Priscilla A. Wold, Pamela K. Allen, Delorse M. Brooks, Beverly Boli, Rita Hysmith, Log Augmentation, Raymond R. Panko, Susan Gail Roetter, Robert Louis Belleville, Ann Weinberg, Adrian C. McGinnis, Robert S. Ratner, David S. Maynard, Robert N. Lieberman, Sandy L. Johnson, James H. Bair, Jeanne M. Leavitt, Rodney A. Bondurant, Jeanne M. Beck, Marcia L. Keeney, Elizabeth K. Michael, Jonathan B. Postel, Elizabeth J. Feinler, Kirk E. Kelley, N. Dean Meyer, James E. (Jim) White, Douglas C. Engelbart, Martin E. Hardy, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman
Jeanne M. Beck, David C. Smith, Beverly Boli, David C. Smith, Jonathan B. Postel, Priscilla A. Wold, Rita Hysmith, Pamela K. Allen, Delorse M. Brooks, Elizabeth F. Finney, Beverly Boli, Lawrence A. Crain, Kirk Sattley, Susan Gail Roetter, Robert N. Lieberman, Ann Weinberg, Kenneth E. (Ken) Victor, Douglas C. Engelbart, James H. Bair, Elizabeth K. Michael, Richard W. Watson, Elizabeth J. Feinler, Harvey G. Lehtman, Kirk E. Kelley, Laura E. Gould, Jeanne M. Beck, Dirk H. Van Nouhuys, James C. Norton, Dirk H. Van Nouhuys, Ann Weinberg, Kirk E. Kelley, Israel A. Torres, Jan H. Kremers, Susan K. Ocken, Raphael Rom, David C. Smith, Buddie J. Pine, Andy Poggio

KIRK 26-OCT-75 21:43 26755

Implementation of procedures for generating useable hardcopy documents from help files

How?

Implementation of procedures for generating useable hardcopy documents from help files

To simply state that Help should be the up-to-date source of ARC hardcopy documention is not going to make it happen; though the benefits that would result from such an approach in maintenance of documentation of a system as volitile as the AKW are obvious. With the same amount of effort currently going into maintance of information duplicated in online and offline forms, we could have up-to-date online and hard copy documentation in all areas instead of the lagging and sporadic coverage we have been suffering. 1

Reaching our stated goals requires the development and implementation of procedures for doing it. Some feasible techniques for writing documentation which will work both online and offline have been developed and applied. For example, see the Readmail design document / help file / userguide <26464,>. This document can be Output Processed to produce a useable Line Printer or COM version. At the same time, the directives can be deleted to produce a file well formatted for use in Help. The Help development document, Helpd, is another such file. 2

As part of my current NLS-8.5 and 9 Help documentation writing, I would like to begin incorporation of some existing hardcopy documentation. Subsequent to being incorporated into help, new editions of a document will be generated directly from the Help file. Note that documents can be generated via output processor directives in any form desireable. The major part of the work I will do is to maintain a good output processor / COM format by structuring the files and adding directives in such a way that when output processor directives are deleted, the document will become a workable help file (assumming the existance of a useable help accessing system). I will also update material obsolete due to changes in 8.5 or 9. 3

Documents that are applicable in my current work with the core AKW help file are "Preface to NLS Tools", "Tenex Guide for Users of NLS", "Lineprocessor users' Guide", "TNLS-8 Primer", "Introduction to DNLS", and "Help Services Sample Session". In the future, I expect to see Base, Sendmail, Programs (including Dean's programming guide), and the loadable subsystems treated in the same manner. The "Publications" help file being written by DVN is going in this direction. It contains the Output Processor Users Guide, among other things. 4

In order for this to work, those people responsible for the updating and maintenance of future versions of these and other documents which are incorporated into Help, must understand and agree to follow the procedures. 5

Essentially, all this means is knowing where the source files are (XHELP) and agreeing to update these files. 6

Implementation of procedures for generating useable hardcopy documents from help files

Since some of the documents involved are maintained by Applications, I'm not sure what I should do to get agreement on these procedures. 7

I AM sure that ARC cannot afford the effort that currently is spent maintaining information duplicated in both hard copy and online formats. Since the implementation of these procedures would involve Applications and Development personnel working together, it has the potential at least of helping to sew up the split that has divided the documentation effort at ARC. 8

KIRK 26-OCT-75 21:43 26755

Implementation of procedures for generating useable hardcopy documents from help files

(J26755) 26-OCT-75 21:43;;;; Title: Author(s): Kirk E. Kelley/KIRK; Distribution: /DAV([INFO-ONLY]) POOH([INFO-ONLY]) JDH([INFO-ONLY]) EKM([INFO-ONLY]) HGL([INFO-ONLY]) BEV([INFO-ONLY]) RWW([INFO-ONLY]) DVN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: KIRK;

26755 Distribution
David C. Smith, Ann Weinberg, J. D. Hopper, Elizabeth K. Michael, Harvey G. Lehtman, Beverly Boli, Richard W. Watson, Dirk H. Van Nouhuys,

26755 Distribution
David C. Smith, Ann Weinberg, J. D. Hopper, Elizabeth K. Michael, Harvey G. Lentman, Beverly Boli, Richard W. Watson, Dirk H. Van Nouhuys,

Documentation Weekly report for week ending 10/26/75

Bev 1

This Week 1a

Made changes in format for '74 Final Report. Sent off for two proof versions from DDSI. 1a1

Worked on Lexicons in Xhelp, Base and Core. 1a2

Edited part of minutes from KWAC meeting. 1a3

Did some directory housecleaning and catching up after week in Boston. 1a4

Kirk 2

Done 2a

Gathered and reviewed Hardcopy sources for Core. 2a1

Wrote a proposal for implementing help --> hardcopy conversion process. 2a2

Fixed bugs Bev found in core. 2a3

Discussed Help with Dav 2a4

Do 2b

Discuss Programs and Cobol status with JAC3. 2b1

Discuss Help with Bev and Dav 2b2

Incorporate Useful Hardcopy into core. 2b3

Establish procedures for help --> hardcopy conversion and maintenance. 2b4

BEV KIRK 27-OCT-75 01:42 26756

Documentation weekly report for week ending 10/26/75

(J26756) 27-OCT-75 01:42;;;; Title: Author(s): Beverly Boli, Kirk E. Kelley/BEV KIRK; Distribution: /ARC-DEV([INFO-ONLY]) ; Sub-Collections: SRI-ARC ARC-DEV; Clerk: KIRK;

26756 Distribution

Jan H. Kremers, Susan K. Ocken, Raphael Rom, David C. Smith, Andy Poggio, David L. Retz, Jan A. Cornish, Larry L. Garlick, Delorse M. Brooks, Beverly Boli, James E. (Jim) White, Ann Weinberg, Kenneth E. (Ken) Victor, Dirk H. Van Nouhuys, Jonathan B. Postel, Elizabeth K. Michael, David S. Maynard, Karolyn J. Martin, Harvey G. Lehtman, Kirk E. Kelley, Charles H. Irby, Robert Louis Belleville, Don I. Andrews, Richard W. Watson, Douglas C. Engelbart,

DAV 27-OCT-75 12:53 26757

Bug in NLS 8 at BBNB

There is a journal link in DCE's Oct 24 message in the line "A Special-Studies Support Center," (24758,), ... When I try to Jump to Link and bug it, I get "system file error", which should probably never happen. But even more peculiarly, when I Jump to Link and TYPE in the link, I get "file not on line...". 1

DAV 27-OCT-75 12:53 26757

Bug in NLS 8 at BBNB

(J26757) 27-OCT-75 12:53;;;; Title: Author(s): David C. Smith/DAV; Distribution: /FEEDBACK([ACTION]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: DAV;

26757 Distribution
Special Jhb Feedback,