NLS: AN OFFICE-ORIENTED INFORMATION SYSTEM

- DOCUMENT PRODUCTION
- INFORMATION MANAGEMENT
- INTERPERSONAL COMMUNICATION
- INTELLECTUAL SUPPORT

AUGMENTATION RESEARCH CENTER
STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA 94025 U.S.A.

FOR ADDITIONAL INFORMATION

East Coast

David A. Potter
163 Hamilton Ave.
Princeton, New Jersey 08540
(609) 924-5126

West Coast

James C. Norton
Augmentation Research Center
Stanford Research Institute
Menlo Park, California 94025
(415) 326-6200 ext. 2124

An Office-Oriented Information System

NLS Catalog Number 25083

NLS: COORDINATED INFORMATION PROCESSING SERVICES

NLS is a comprehensive set of computer-based aids for people who work with information. NLS services, available nationwide on time-shared computer facilities, include text processing, message handling, document production and photocomposition, graphics integrated with text, information storage, management and retrieval, and much more. These features reflect nearly fifteen years of continuous design refinement and hundreds of thousands of hours of online user experience. This brochure is an introduction to NLS that describes how the system can improve your effectiveness.

NLS: SERVICES FOR PEOPLE

People are the key element in the NLS world. NLS has been developed with just one aim: to improve the information handling capability and effectiveness of people. People use it to coordinate projects, formulate and communicate decisions, collaborate, enter data, generate everything from letters to proposals, publish documents, and organize their daily work routines. It is employed by scientists, clerks, managers, typists, editors, researchers, programmers, writers, and secretaries. Geographically dispersed groups of co-workers use it for collaboration.

In short, NLS facilities are used to help people generate, manipulate, store, retrieve, and distribute information. NLS applications are as diverse as information needs and uses. All the NLS services taken together provide a complete environment where information processing work is performed.

The following description of some NLS application areas illustrates the wide range of services available.

APPLICATIONS SUPPORTED BY NLS

Text Handling

* a new dimension of editing power

NLS includes an extremely versatile and powerful text editing system. The advantages of NLS editing over other automated editing systems are based on two unique structural features.

The NLS Statement. Two basic logical units of written English are the paragraph and heading. The paragraph and heading are the natural units of information for text manipulation because they represent ideas. Most computer editing systems ignore these logical divisions in prose, providing the user with a line-oriented editor which operates on sequential files. In such systems, the line is the basic unit. A line, however, is a totally arbitrary block of text, which does not correspond to any natural information unit; thus, editing systems that manipulate "lines" do so for the convenience of record storage devices, disregarding the convenience of the people working with the text.

NLS is designed for the convenience of the people using it, not the machine. The basic unit for information storage is the NLS statement, a logical block of text. It may be a paragraph, a diagram, a chapter heading, or a row in a table. Regardless of its function, the NLS statement bears a one-to-one correspondence to the ideas expressed in the document.

The NLS Structured File. The structure of information is important to NLS. Text files in NLS are built up in outline form. For example, chapter headings may form the highest level, sub-chapter headings a second level, etc. The NLS user generally creates a file structure that corresponds to the natural hierarchical structure of his ideas. The structuring of NLS files greatly simplifies paragraph or chapter manipulation. Editing operations which are considered "heavy" for most editors are routine for NLS.

These structural features of NLS result in greater editing power and flexibility, in a system so natural that anyone can easily learn to use it.

* flexible information retrieval and display

NLS provides special features for online information retrieval and display. Users control the content and format of the information that they read. With the aid of built-in or user-supplied "filters," the NLS user can easily select and browse through any online information.

Online Browsing and Retrieval. The outline structure of NLS files or data bases adds a new dimension to online retrieval and browsing. For example, simply looking at the headings or top two "levels" of a document gives the user a quick table of contents of the document. This level "clipping" and browsing operation may be repeated to locate information or to explore any document at any time.

Combining this unique NLS "browsing" facility with more standard pattern matching or logical retrieval functions provides the NLS user with extensive retrieval and display facilities.

Multiple Displays. Because NLS is not based on "lines" of information, it is possible to use NLS with a variety of display or hardcopy terminals. Moreover, the user can easily partition the screen into two or more independent display areas. Each area of this "split" screen may be independently controlled, permitting concurrent viewing and manipulation of several files.

This multiple display capability greatly simplifies the construction of new documents from sections of existing ones. Cross-file and multiple-file editing, which are often difficult and tedious with other editing systems, are simple with NLS.

* extremely high security and integrity

Protection Against Loss. All files are protected from both accidental user errors and hardware failures. If the system "crashes", there are copies of work in progress on disks, and if something is accidentally deleted, a user may always return to a previous copy (either online or from tape storage).

Privacy. Access to every individual file may be limited by the owner of the file to a particular individual or group. In the case where additional protection is desired, information may be encrypted so that only those with the "key" may ever read it. When files are shared, NLS maintains a "signature" for each statement that records the name of the person who made the last changes to the statement and the time the changes were made.

Automatic Backup and Archiving of Information. Files are automatically archived onto offline tape if they are not accessed for a certain number of days. This provides automatic "house cleaning" for users, but is only an option. A record of all archived files is maintained and a user can retrieve any archived file with a simple command. This means that information, once entered in the system, is never lost: it is available for retrieval and immediate use after months or even years.

* additional facilities

Online and Offline Information Entry. Text and other information may be entered from online terminals or may be recorded on tape at terminal for later entry into NLS. Tapes can be read into the system to be converted into online form automatically.

Choice of Display or Hardcopy Terminals. NLS supports a wide range of standard, off-the-shelf display and teletypewriter terminals. The teletypewriters may be small portables or high-quality printers.

Forms Production. Since forms play an important part in an organization's information processing, NLS includes easy to use facilities for creating preprinted blank forms or forms that may be filled in online and printed.

Automatic Spelling Correction. Any NLS file may be checked for misspellings; those words that appear in a 50,000 word dictionary will be automatically corrected. Users can supplement the main dictionary with their own personal dictionaries.

Document Production

* fully integrated system

  Text and line graphics can be mixed and used together in the same document; they can also be "proofed" on a high-quality display terminal in the format that will appear in the final printing. Thus, the user is able to see the drawings and text in their respective positions on a page before photocomposition.

* choice of output devices

  In addition to hardcopy terminals, the system can produce documents on high speed printers and photocomposition devices.

* versatile, user-controlled formatting

  The user selects and inserts formatting instructions to control page layout, justification, hyphenation, character size and font, etc. The system can handle almost all of the most complex document production problems.

* automatic formatting

  A library of standard formats is available that may be applied to a document, or users may design their own format for particular needs.

* document retrieval and control

  Document retrieval and control facilities automatically index, catalog, and distribute documents that then can be searched by author, title, keyword, and accession number.

* This brochure was typed and edited online, formatted, and printed using the NLS document production system.

Communication and Online Collaboration

* message handling

  Messages or documents may be sent to any number of recipients, in the same office or on the other side of the country. They

may be sent to individuals, or to a preselected distribution list (only the name of the list must be specified by the user). The sender may note that the item requires immediate action or is passed along for information only. Copies of the message or documents sent may be retained and stored by both the author and recipients. Delivery and notification take place within seconds.

* online system for recording, cataloging, and distributing information

Large documents or short messages may be sent to individuals, groups, or organizations, even if the recipients are not using the same computer, and recorded permanently in a special "dialogue record". This record provides for complete privacy or public availability.

All items contained in the dialogue record may be retrieved any time after original distribution using automatically generated keyword indices, accession numbers, and catalogs. Any item may include references to any other online item; readers can use these references to automatically retrieve related documents of interest to them.

This is an ideal medium for coordination or collaboration among geographically dispersed authors, groups, or organizations. In our experience, the savings in time and resources have been dramatic.

* online "linking" of terminals and teleconferencing

Everything that is viewed or printed at one terminal can appear simultaneously on another, permitting the instantaneous sharing of information. This powerful new communications medium permits people at different locations to work together more easily, often reducing the need for travel.

Advanced Design Features

* user profile

Each user group and each individual user has unique requirements, expectations, and preferences. NLS adapts to these needs by maintaining a "profile" for each user. The profile "remembers" things such as the degree of prompting the user wants and the way the command language should appear This al-

lows users to "customize" NLS to meet their own use styles or levels of expertise.

* online help and question answering facilities

Infrequent or novice users of any system experience difficulty in remembering how to do things with it. Recognizing this problem, we have built into NLS a simple yet comprehensive online help facility. The user may type "?" at any time to see a list of the things that may be done at that moment. If this does not suffice, the user may also push a "HELP" key at any time, which automatically invokes an online question-answering system that explains all the NLS commands, concepts important in understanding NLS, how to perform certain tasks, and other related information. Even advanced users have questions about NLS which this online instruction facility can answer.

* modular design, extensibility

NLS contains many integrated command modules, called subsystems. Each subsystem serves a particular set of information handling needs. This modular approach makes it easy to customize NLS to fit the unique applications needs of user groups, and permits new facilities to be added very easily by SRI or by the users themselves. NLS can also be connected to other computer systems to take advantage of existing specialized capabilities.

THE NLS UTILITY SERVICE

NLS is available through subscription to the Utility Service run by SRI. Subscribers pay for a share of computing power and information storage, along with support service that includes training, application consulting, documentation, programming support and operator services. Many organizations currently subscribe to this service. A key person from each organization, called an Architect, works with SRI to plan the use of NLS in his or her environment.

It is very important that an integrated office automation system include user training and applications consulting by specialists. In our experience, personal attention is necessary for efficient assimilation of this new technology. Our SRI staff members are experts at helping you enter the NLS world and are always available for training and consultation.

DVN 7-MAR-75 14:34 25084

Rough NSW Documentation Hopes

Sent with inclosures via US mail 3/7/75

Rough NSW Documentation Hopes

Augmentation Research Center
Stanford Research Institute
Menlo Park, California 94025

Stephen Warshall
Massachusets Computer Associates, Inc.
Lakeside Office Park
Wakefield, Massachusetts, 01880

Dear Steve:

1

As you may know, we are without a printer until our operation on the net shakes down a little more so I take the liberty of sending you this somewhat ugly copy. I am also journalizing this letter. 2

Following our phone conversation of february 26 I enclose examples of the kind of documentation we hope to see for the Frontend and Works Manager. For each of the tools for which we are responsible, we plan to have an online data file suitable for query by the NLS Help system, a command summary, and one or more tutorial scenarios and discursive prose introductions as needed. (hjournal,25395,) contains our present division into tools with the appropriate documentation; this outline may change if the boundaries between tools change. It further seems appropriate that there should be an overall NSW cue card in which the Frontend and Works Manager commands would take an important place. 3

I enclose an example of a Command Summary, the NLS-8 Command Summary, a tutorial scenario, the TNLS-8 Primer, a cue card, the NLS-8 cue card, and excerpts from the online file that supports queries of selected NLS-8 commands. 4

These excerpts from the Help data base require some explanation. I understand Susan Roetter will be in Boston next week to train people at MCA in NLS. She will show you how Help operates online: a user types in a term or phrase and the system responds with an explanation and a menu of related topics. 5

samples nodes in the data base contain information of one of 5 types: 6

explanations of commands (which include the command syntax as the first line) 6a

(green on the enclosed samples), 6a1

special effects of commands (always substatements under the explanations of commands) 6b

(orange in the samples enclosed), 6b1

examples 6c

(red), 6c1

explanations of concepts 6d

(black), 6d1

and links that make nodes somewhere else in the files available as menu items 6e

(blue). 6e1

I here include examples of a simple command (Delete File(1), instruction in getting Help (2), a complex command (the Output group(3) and a section where there are many explanations of concepts (4). Susan Roetter has with her a prototype copy of the NLS glossary which we intend to publish shortly and which consists of explanations from the glossary arranged alphabetically and reformatted as appropriate for hard copy. 7

There are some more details some one must understand in order to make a file that functions properly with the query system, but what I have enclosed seems enough for you to get started. <xprograms,helpd,help,> at BBN gives a complete account. 8

Time is flying and we need to keep in touch and keep moving on the matter. We must for example settle a list of frontend and Works manager commands that the user will see, before the end of the month. <Hjournal,25383,> gives a good beginning. I hope to hear from you soon. 9

Rough NSW Documentation Hopes

yours sincerely, Dirk

10

Sincerely,

Dirk H. van Nouhuys
Augmentation Research Center

DVN 7-MAR-75 14:34 25084

Rough NSW Documentation Hopes

(J25084) 7-MAR-75 14:34;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION] dirt notebook please) DIRT([INFO-ONLY]) MCG([INFO-ONLY]) WEC([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT; Clerk: DVN; Origin: < HAMILTON, WARSHALL,NLS;2, >, 7-MAR-75 13:42 DVN ;;;;
####;

10 JUNE 1975

Proposal For Research
SRI No. ISU 75-94

CONTINUATION OF NATIONAL SOFTWARE WORKS DEVELOPMENTS

Part One---Technical Proposal

Prepared for:

Information Processing Techniques Office
Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia

Attention: William E. Carlson

Prepared by:

Richard W. Watson,
Assistant Director
Augmentation Research Center

Approved:

Douglas C. Engelbart, Director
Augmentation Research Center

Bonnar Cox, Executive Director
Information Science and Engineering Division
Stanford Research Institute

Proposal For Research No. ISU 75-94
Continuation Of National Software Works Developments

Table of Contents

SUBJECT: 2

CONTINUATION OF NATIONAL SOFTWARE WORKS DEVELOPMENTS 2a

This proposal by the Augmentation Research Center (ARC) of SRI covers tasks for the National Software Works (NSW) Program during the nine-month period, July 18, 1975 through April 17, 1976. The developments proposed are a consistent extension of the work presently in progress under Contract No. F30602-75-C-0156, and it is expected that acceptance of this proposal would lead to an extension of that contract. 2b

The National Software Works Program has as one goal, creation of a framework for sharing computer resources. This framework is based on a communication network and constructed to support tools that aid every phase of software development. Its framework is designed to promote the easy integration and coordination of software tools and the gradual strengthening of their cooperation into a unified system [1,3,5]. (References are listed at the end of Part One of this proposal.) 2c

The initial version of the NSW will become operational for prototype use in the summer of 1975. There is yet much to be done toward the goal of delivering a polished final product in July 1978. 2d

The tasks proposed here are of two main types: tasks that further the construction of the basic NSW system framework, and tasks that provide tools for operation within the NSW. 2e

There are four main components of the NSW system: 2f

1) An NSW Frontend system (Frontend) that provides access through terminals to a communication network (ARPANET initially) and provides a set of services that help create a consistent NSW user environment. 2f1

2) An NSW Works Manager that provides a coherent environment for the user, an NSW file system, resource allocation and control, accounting, authentication, and so forth. 2f2

3) Protocols and conventions needed for communications and control between the various components of the NSW system, Frontend and Works Manager, Frontend and tools, and Works Manager and tools. 2f3

4) The tools and services (hardware and software) that will work within the NSW environment. 2f4

OBJECTIVES: 3

I SOLIDIFICATION OF THE NSW FRONTEND SYSTEM 3a

Work on this objective will involve improvements in the area of efficiency; support of new terminal types; system, tool supplier and user documentation; system shakedown and integration with other NSW components; evaluation of the Control Meta Language used to specify the user interface; consultation with tool suppliers and provision of a cross network maintenance environment. One top priority objective will be to assure smooth use of the Frontend with previously existing tools built outside the NSW environment. Another objective will be to add disk storage with the goal of being able to support around 20 simultaneous users on the PDP-11/40 configuration. 3a1

II CONTINUED EVALUATION OF THE NSW PROTOCOLS 3b

The top priority tasks will revolve around shakedown and integration of initial protocol implementations with other NSW system components into a solid reliable NSW system. Some implementation changes may be necessary to improve efficiency and to provide error and crash recovery. 3b1

Work on this objective will also involve incorporation of the experience gained by ARC and others, during development and implementation of the initial NSW system, into the designs for the next versions of the basic NSW communication and process control protocols and application protocols. New design features are expected to be needed to support improved efficiency, crash and error recovery, and resource sharing of multiple copies of the Works Manager, Frontends, tools, or files. An important part of this process will be to achieve consensus on NSW protocol designs with the community of those implementing them on various machines and using them. 3b2

Documentation work will consist in fully documenting the initial protocols as finally implemented, collaboration with MCA on a Tool-Bearing-Host suppliers manual, and design documentation of new designs. 3b3

III PROVIDE NEW AND IMPROVED TOOLS 3c

We will focus on providing close support and evaluation of the new NLS graphics facilities, consulting to the NLS user community at Gunter AFS, complete integration of NLS into the NSW environment, and other enhancements to the NLS family of

tools seen useful by the NSW user community, with prime emphasis on documentation facilities. 3c1

BACKGROUND AND TECHNICAL NEED: 4

The need for the NSW is described in references [1,3,4]. 4a

ACCOMPLISHMENTS: 5

ARC's general background is summarized in reference 5. Within the NSW, ARC has been playing an active role not only in its specific tasks as defined in reference 5, but also in overall NSW design and integration. 5a

Specific accomplishments in each task area are described later in the next section. 5b

PROPOSED EFFORT: 6

(Manpower estimates given below do not include supervision or clerical support. These are estimated in the cost totals given in Part II. The estimates are made subject to the uncertainties listed in the Statement of Work.) 6a

I NSW FRONTEND DEVELOPMENT 6b

GENERAL OBJECTIVES OF THE NSW FRONTEND 6b1

The Frontend provides the logical or conceptual function of interfacing a user to the NSW (Works Manager and tools). It consists of two sets of processes which form a logical whole. One set is to run on a satellite computer system local to the user. The other set is to run on other hosts within the NSW. The decision as to which set best suits a particular function will change over time as computer hardware and communications costs change, and as the NSW evolves. The initial Frontend consists of a PDP-11 satellite, providing terminal control and command language services, and a Help system running on a remote PDP-10. The combination of services provided by the Frontend and Works Manager are a logical whole as seen by the user and constitute an NSW Executive. The Frontend is to provide the following types of services to the NSW: 6b1a

1) To provide the user with a coherent and consistent command language and command portrayal discipline throughout the NSW. 6b1b

No matter whether the user is giving commands to a tool (a tool is a software service such as an editor, compiler, calculator, etc.) or to the Works Manager or the Frontend itself, he does so using the same language and control conventions for specifying which commands he wishes executed and the same conventions for specifying arguments or parameters to commands; he performs editing of commands and literals with the same conventions, and gets the same type of prompting, and requests help in the same way. The general syntactic form(s) should be the same from tool to tool unless there is good reason for the tool to deviate from the standard. Of course the particular commands and vocabularies will vary with the tool, and in fact the same verbs may be used with quite different semantics in different tools; but at least most other facets of the command language discipline (including asking for help and being prompted for the

proper type of input) should be consistent across tool boundaries. 6b1b1

It is expected that initial users of the NSW will have to access some tools in a mode (which we will call transparent mode) where the Frontend cannot provide the user with the full range of facilities just described. This would be the case for some tools built outside the NSW framework and inserted with minimal changes. However, we expect that as time goes on and the NSW grows, the user will be able to use most tools through the unified user interface provided by the Frontend. 6b1b2

2) To provide tools with well-formed commands 6b1c

This is being accomplished by issuing remote procedure calls to "external" procedures in the tools to actually execute commands. This will be implemented through the Distributed Programming System (DPS) being developed under ARC's protocol task below. 6b1c1

Many operating systems and application programs have elected to use half duplex, line-at-a-time terminals because of the increased computer and communications efficiency provided by this approach. Other operating systems and application programs have chosen, instead, to use character-at-a-time full duplex terminal disciplines because of the opportunity this provides for using a more human-engineered command language discipline. 6b1c2

The NSW Frontend combines these two approaches into a COMMAND-AT-A-TIME system, where the application programs do not directly interact with the terminal (for tools using the full set of Frontend services), but rather receive fully specified commands from the Frontend. At the same time, the Frontend will attempt to provide the user with the best possible human-engineered command language discipline. 6b1c3

3) To provide a terminal-independent interface to the tools 6b1d

Because the Frontend handles all terminal interaction (except in transparent mode), it will present to the tool a virtual terminal. Thus, once a tool is developed, little attention need be given to the type or particular characteristics of the terminal the end user may choose to employ while using the tool. In fact, the cost of

creating new tools should be considerably reduced because of the facilities made available by the Frontend. 6b1d1

This means that even though the creators of a tool envisioned the user sitting at a typewriter terminal, the NSW user who happens to be using a display terminal with a pointing device may be able to interact with the tool in a two-dimensional sense, for example by pointing to arguments on his screen instead of typing them. 6b1d2

For tools that wish to make more extensive use of a display terminal if the user has one, the Frontend presents primitives for allocating windows on the display and allows the tool to write/delete/move/make invisible items displayed within the windows. 6b1d3

4) To provide tool interaction and coordination facilities 6b1e

The Frontend in cooperation with the Works Manager can provide facilities for the user to easily perform cross tool database communication, parallel tool execution, and automatic sequencing of tool execution. 6b1e1

5) To provide NSW-wide macro facilities 6b1f

The user should be able to specify new processes that are constructed from more basic NSW commands or services and get these executed by expansion and interpretation by a Frontend process. 6b1f1

6) To provide standard mechanisms for error and crash recovery and for presenting status or error conditions to the user 6b1g

There are many points in a multi-host, multiprocess, communication based system such as the NSW where errors or component failures can occur. It is important that the system architecture and other conventions minimize error occurrence and its impact on the user. 6b1g1

7) To provide passive and active help and tutorial facilities 6b1h

Besides providing levels of prompting, it will be useful to provide online documentation and help information for tools accessed in a standard way. It will also be desirable to provide active tutorial facilities for instruction to the user. These services would also

provide a protected environment for a user to experiment with tools and other NSW services. 6b1h1

8) To provide facilities to allow the user to tailor certain Frontend and other NSW characteristics 6b1i

The user or someone acting in his behalf should be able to tailor certain aspects of terminal control, command language, and names in information structures. Alternatively, a process may tailor these parameters on his behalf on the basis of terminal type, his background, task, frequently used tools, etc. 6b1i1

9) To provide terminal linking facilities 6b1j

The ability of two or more users to connect their terminals (especially displays) together in order to work together has been found very valuable. 6b1j1

10) To provide state saving and undo and redo facilities 6b1k

After executing some number of commands users occasionally realize that they have made a mistake and would like to back up to a previous point. Often the user wishes to redo a command or command sequence by performing some operation simpler than respecifying commands. Provision of this type of facility requires both capabilities in the Frontend and agreement by tool builders and suppliers to follow well specified state saving and backup conventions. 6b1k1

11) To provide user statistics for evaluation and analysis groups and adaptive or tutorial programs 6b1l

The Command Language Interpreter of the Frontend is a central point that easily can gather various classes of statistics on user errors, commands, and various types of timing such as command specification time, response time, and execution time. These statistics can be sent to a file for later processing by analysis programs or, in real or background time, could be processed by processes associated with tutorial facilities or used to modify characteristics of the system in the database (User Profile) containing parameters tailored to the user. 6b1l1

ACCOMPLISHMENTS 6b2

1) A Frontend system has been designed to operate on PDP-11 and PDP-10 computers with the following features: 6b2a

a) An operating system interface. 6b2a1

b) Display and terminal control. 6b2a2

c) Appropriate ARPANET protocols. 6b2a3

d) Command Language Interpreter driven by Command Language Grammar and User-Profile databases. This interpreter can support both tools built explicitly within the NSW environment and tools built outside the NSW environment. On July 1 the PDP-10 version will be operational with some work still needed on the display controller. The PDP-11 version will be operational in the Fall. 6b2a4

e) The Frontend provides initial services in categories 1-3, 6, 8, 11 above. 6b2a5

2) A formal language for specifying NSW tool and NSW system user interfaces was designed and a compiler implemented for it to produce the grammar databases above. We call this language Control Meta Language (CML). 6b2b

3) User documentation 6b2c

User documentation in the form of an online HELP database and command language reference manual will exist. We are working with MCA to produce these combining Frontend and Works Manager commands into the NSW Executive as seen by the user. 6b2c1

4) System design overview documents for various aspects of the Frontend exist but these are not integrated into a single document as desirable. Commented source code listings will be available. 6b2d

5) Integration of the Frontend with other NSW components will have begun, but will not be completed at the end of the current contract period. 6b2e

6) System support tools in the form of an initial low level debugging environment from a PDP-10 to PDP-11, and cross compiler from the PDP-10 to PDP-11 for the L-10 high level

system programming language were developed. The L-10 language was extended to support features needed for Frontend and protocol implementations. For example, a coroutine linkage has enabled us to implement the Command Language Interpreter in 3-4 times less memory space than a previous related interpreter. This is very important in the memory space limited PDP-11. 6b2f

7) Active participation in the overall NSW design was carried out. 6b2g

8) As the first major user of the ELF operating system for the PDP-11 we have been heavily involved in its debugging and stabilization. 6b2h

SPECIFIC OBJECTIVES FOR THIS PROPOSAL 6b3

1) The Works Manager (WM) user interface will be integrated with the WM on the same and different PDP-10 Tenex systems. This task requires delivery of a complete and debugged (as much as this is possible) Command Language Interpreter (CLI) and Control Meta Language (CML) and L-10 compilers. CML and L-10 compilers will run as NSW tools on a PDP-Tenex with an NLS source language environment. Necessary implementations of NSW protocols to support FE WM intercommunication are also required for this task (see Protocol tasks). 6b3a

2) It will be a goal to support 20 users simultaneously on a PDP-11 Frontend. This will require use of disk storage, as the PDP-11 does not have adequate main memory capacity to support that number of users. 6b3b

3) ARC will assume responsibility for refining the release of virtual memory (VM) ELF available from Speech Communications Research Laboratory (SCRL) into an operating system which can support the NSW FE software. The version of VM ELF containing all the needed features for the NSW FE is not scheduled for release by SCRL until August. This release will allow loading from the network of FE software as a user program. Earlier releases may prove useful and will be used as appropriate. ARC will accept responsibility for debugging code already written by SCRL that utilizes disk storage for virtual memory. 6b3c

4) A Command Language Interpreter will run under ELF as a user program and communicate appropriately with WM and NSW tools. The CML compiler and associated post processor will produce grammar data structures for the PDP-11 version of

the FE. An L-10 cross compiler and run time environment will exist from the PDP-10 to PDP-11. 6b3d

5) A cross network debugging environment from PDP-10 to PDP-10 or PDP-11 for FE maintenance will be produced. 6b3e

6) Three terminal classes will be supported by the FE: half-duplex line-at-a-time typewriter terminals (possibly not supported by ELF on the PDP-11), full duplex character-at-a-time typewriter terminals, and two-dimensional CRTs using the ARC developed Line Processor. These classes will be supported on a PDP-10 Tenex through a TIP. 6b3f

7) We will collaborate with other contractors in crash and error recovery analysis and design of appropriate strategies. Some implementation changes may be required during this contract period. 6b3g

PROPOSED FRONTEND (FE)TASKS 6b4

The specific tasks necessary to meet the above objectives are outlined below. 6b4a

1) ELF OPERATING SYSTEM SUPPORT (9.5mm) 6b4b

General Support (3.5mm) 6b4b1

The ELF operating system was chosen by the NSW Steering Committee for the initial Frontend system. We had expected to receive delivery of a fully debugged system requiring little special attention on our part. Experience has indicated that that assumption was overly optimistic, and some effort on our part has been required to create a stable system. 6b4b1a

We would expect from this experience that some ongoing effort in the area of general bug fixing and dealing with problems in the just released Virtual Memory ELF (VMELF) and liaison with its developers at Speech Communication Laboratory (SCRL) will be required over the time period of this proposal. 6b4b1b

Design and Implementation of a Network Control Program (NCP)/ELF File System Interface (1mm) 6b4b2

This task involves implementation of an interface package to allow user programs in ELF to communicate

with the ARPA network. In ELF, communication with the ARPANET is controlled by a centralized set of processes which are referred to as a Network Control Program (NCP). User programs in ELF run in protected areas of memory (virtual address spaces). A "front-end" must be added to the ELF NCP to allow the NCP to receive calling parameters from user virtual address spaces. This capability is not currently provided in ELF, but is required for implementation of the ELF File Transfer Protocol package, Distributed Programming System, and application protocols for the PDP-11. 6b4b2a

PDP-11 Disk (5mm) 6b4b3

The exact number of simultaneous users that the initial version of the NSW Frontend will be able to support is not yet known precisely; it is expected to be around 10. It is expected that the limiting resource will be core memory size on the PDP-11/40 rather than CPU cycles. By July 1975 the ELF operating system will contain unchecked-out code to support an extension of PDP-11 address space using disk memory. 6b4b3a

No other group has funding to complete debugging and integration of the Virtual Memory disk support code. 6b4b3b

The objective of this task will be to add disk memory and debug the code mentioned above with the goal to support at least 20 simultaneous users. 6b4b3c

2) CONTINUING INTEGRATION WITH WORKS MANAGER AND TOOLS (3mm) 6b4c

Integration of the various subsystems of the initial NSW is expected to begin about June 1, 1975. This work is not expected to be completed during the current contract period, and some additional effort will be required to get all components communicating appropriately with each other. 6b4c1

3) TASKS NECESSARY TO INSTALL FRONTEND ON A PDP-11 (3mm) 6b4d

Tasks necessary to install the NSW Frontend on a PDP-11 are the following: 6b4d1

Interface of the L-10 run-time routines with Virtual Memory ELF (VMELF) 6b4d1a

Interface of the Command Language Interpreter and typewriter terminal control with VMELF 6b4d1b

Interface of the Command Language Interpreter with ELF Telnet subsystem 6b4d1c

4) SYSTEM SHAKEDOWN (3mm) 6b4e

As the initial integrated NSW system is put into operation, usage will uncover problems that cannot be easily anticipated at this point. It will be particularly important that any problems associated with support of existing tools built outside the NSW environment be given high priority. 6b4e1

5) ERROR AND CRASH RECOVERY (2mm) 6b4f

There are several major components in the NSW system. Each can fail or generate errors of many types. 6b4f1

Failures in host hardware and operating systems
Failures or errors in tools
Failures or errors in the NSW Frontend in any of its components
Failures or errors in one or more copies of the Works Manager
Communication Network problems or breaks 6b4f1a

We need to work closely with the appropriate other NSW contractors and both enumerate each type of expected error or failure, and specify alternative method as for general communication and control, state saving,and recovery procedures for dealing with these. The recovery procedures will be specified in such a way that, whenever possible, no intervention is required by the user and, if possible, recovery is invisible to the user. The alternative strategies with their associated costs will be presented to the NSW Steering Committee for selection. 6b4f2

In particular, in the event that the Frontend crashes, on recovery it needs to obtain from the Works Manager such information as a list of active users and a list of the tools they were using, grammars for the tools, appropriate communication path connections to the tools, user profile databases for the active users, connections to help processes associated with tools, and possibly terminal characteristics. 6b4f3

The objectives of this task will be to complete the analysis described above, perform any minimal ad-hoc changes in the implementation needed to deal with major operational problems, and complete design changes for the next iteration on the Frontend. If more extensive ad-hoc changes are required, then other tasks may have to be slipped. Guidance of the NSW Steering Committee will be sought as appropriate. 6b4f4

6) TERMINAL CLASSES AND DEVICES TO BE SUPPORTED BY THE NSW FRONTEND (4mm) 6b4g

The SRI software system available at the end of the current contract will support typewriter type terminals of both the character-at-a-time/full duplex and line-at-a-time/half duplex varieties. Work will be required to smooth the user interface to the latter variety and complete implementation and integration of the display controller for the two-dimensional terminals with a pointing device. 6b4g1

Our goal is to provide an interface between the ELF operating system and command language interpreter that isolates the command language interpreter and its associated terminal support from detailed knowledge of specific terminal characteristics. That is, the FE software should only have to know how to deal with a small number of generic terminal classes. The detailed drivers to handle specific terminals would exist at the ELF level and character streams in specified formats for generic terminal classes would be passed to the FE software. This approach will reduce the cost of adding new terminal types. We will produce a driver to support terminals connected to the SRI Line Processor. Presently the PDP-11 ELF operating system does not support line-at-a-time half-duplex terminals. To support them will require device drivers, modifications to the ELF Executive and ELF Telnet. The total effort involved is estimated to be between 2 and 3.5 man months of effort beyond the estimate above. If these terminals are desired to be supported by ELF a separate add on to the contract resulting from this proposal should be negotiated later. 6b4g2

7) EFFICIENCY AND MEASUREMENT (3mm) 6b4h

Efficiency is important if the goal to support at least 20 simultaneous users is to be achieved. 6b4h1

There are three components of the efficiency question: efficiency of system resource utilization, efficiency of users in getting their work done, and efficiency in the use of available memory space. There are tradeoffs available between these components in terms of cost and choice of system parameters and algorithms. 6b4h2

We will perform measurements on the various software components of the Frontend system to determine how the system components are being utilized and make whatever changes are appropriate to improve systems utilization efficiency while maintaining a subjectively determined level of system responsiveness. One measurement study will be to compare timings of running of NLS in its present form with its NSW form utilizing the NSW Frontend. 6b4h3

As indicated above, the number of terminals that a given PDP-11 can support is expected to be memory space limited. An important part of this task will be to determine what steps can be taken to more efficiently utilize main memory space without serious impairment of performance goals. 6b4h4

8) DOCUMENTATION (7mm) 6b4i

There are five types of documentation required. 6b4i1

1) System Documentation (2mm) 6b4i1a

Documentation for the system will include commented source code files and system overview descriptions. 6b4i1a1

2) Tool Supplier Documentation (1mm) 6b4i1b

Documentation for tool suppliers will include a) descriptions of Frontend services available to tool builders, and b) documentation on CML and its derivatives. We will work with other NSW contractors to integrate this material into a tool suppliers handbook. 6b4i1b1

3) Hardcopy user documentation (1.5mm) 6b4i1c

Hardcopy user documentation will include our contribution to joint Frontend/Works Manager user guides, command summaries, and cue cards. 6b4i1c1

4) Online user documentation (1.5mm) 6b4i1d

Online user documentation will consist of Help database entries for Frontend user features and commands. 6b4i1d1

5) Technical reports and study documents as relevant. (1mm) 6b4i1e

9) AN NSW DEBUGGING ENVIRONMENT WITH SPECIFIC DEMONSTRATION IMPLEMENTATION (5mm) 6b4j

Introduction 6b4j1

It is estimated that at present 1/3 to 1/2 of the development time of programs is spent in debugging them. Further, ongoing maintenance costs generally exceed initial development costs. There has been much development and experience in the debugging area within the R&D community but this experience is generally not available to DoD programmers. 6b4j1a

The NSW offers an excellent opportunity to create a coherent and consistent debugging environment across tools and languages that makes these new techniques widely available. Further, it is important to provide a debugging framework that allows new special computers for which software is being debugged within the NSW to be easily inserted. Because of its importance and the fact that this area may be unfamiliar to some readers, the general approach being considered is discussed in some detail in Appendix A. 6b4j1b

A sound debugging environment is needed to allow the NSW Frontend, NLS Backend, and Works Manager Backend to be remotely maintained. 6b4j1c

The approach described in Appendix A not only meets the operational and developmental needs of the NSW, but can serve as an example of what is possible in the future for DOD applications developed in the NSW environment more generally. 6b4j1d

In summary what is being proposed is the following: 6b4j1e

1) Specification of protocols and conventions for interactive debugging in the NSW environment. 6b4j1e1

2) Specification of the features needed in a debugging facility and language constructs for their implementation. 6b4j1e2

3) Implementation of the above for the L10 Language to facilitate maintenance of the NSW Frontend and to serve as an example of the full power of the interactive remote approach being taken. 6b4j1e3

10) EVALUATION OF CONTROL META LANGUAGE (4mm) 6b4K

The Control Meta Language (CML) is the high level language [7] to be initially provided for specifying the user interface for NSW tools. CML provides basic constructs and vocabulary for dealing explicitly with user interface functions such as obtaining a screen selection or command word from the user and feeding information back to the user. A CML program compiles into a data structure we call a Grammar that drives a Control Language Interpreter (CLI) in the Frontend. 6b4k1

There are two areas of continuing work required on CML: 6b4k2

1) Evolution of its structure and functions to meet possible needs of new or old tools that have user interface features that might be difficult to express in CML as presently implemented. 6b4k2a

2) Provision of an even higher level facility that would simplify tool installation and user interface specification by allowing the interface to be expressed simply in command syntax (with feedback) and the appropriate tool execution functions to call when the command is recognized. 6b4k2b

We feel that the CML approach is an important contribution to the NSW and interactive system design generally, and that it needs further evolution based on NSW experience before it will have reached a "product" level of maturity. 6b4k3

The easier we can make it for tool builders and installers to use the services of the NSW Frontend, the sooner the NSW will truly be the desired coherent and consistent user environment. 6b4k4

During the proposal contract period we will determine what design changes need to be made, based on: 6b4k5

1) NSW experience to date
2) experience in working with initial NSW tool suppliers
3) criticism, evaluation, and feedback from other language designers and potential tool suppliers. 6b4k5a

Implementation of design changes will not take place

during this contract period except for solving urgent problems which might appear. 6b4k6

11) PLANNING FOR NEW FE FEATURES (1mm) 6b4l

Working in collaboration with the TCC, initial users, and others prepare a recommendation for a new user and system features for the FE. In particular, we will work with BBN in the NSW installation of their NLS-Scholar tutorial system to see how it and the FE might more closely cooperate [8]. 6b4ll

DELIVERABLES 6b5

Deliverables will be: 6b5a

1) Source files of debugged software to support the above described capabilities on a best effort basis for PDP-10 and PDP-11 computers. 6b5b

2) Documentation as listed above. 6b5c

CONCLUDING REMARKS 6b6

The focus of the proposed tasks is on providing a stable system for initial users with the main frontend features needed to begin exploring the NSW concept; needed documentation, and some evaluation and planning needed as preparation for the work to follow. The proposal for work to follow this effort is expected to focus more on new features needed to provide facilities in all areas described in the General Objectives to achieve a logically complete Frontend. 6b6a

II NSW PROTOCOLS 6c

GENERAL OBJECTIVES AND BACKGROUND 6c1

The NSW protocol work must meet the needs of a system with the following general characteristics: 6c1a

1) Tools, Works Manager, Frontend built on diverse hardware and operating systems. 6c1a1

2) The various components of the NSW geographically distributed, including file storage and processing capabilities. 6c1a2

3) All resources such as tools and files centrally controlled by the Works Manager. 6c1a3

4) The various distributed system components in close cooperation. 6c1a4

5) It should be possible to create or add new tools for the NSW at low cost. 6c1a5

6) Tools built to operate outside the NSW environment should be insertable in the NSW. 6c1a6

Much more specific requirements are imposed upon the Protocols by services needed and supported by Works Manager, Frontend, Tool Bearing Hosts, and Tools. 6c1b

Some example needs follow: 6c1c

The Works Manager needs to be able to move, rename, and perform other manipulations of files to/from/on remote hosts; login and start and stop jobs; and obtain accounting information. After starting a tool the Works Manager must then be able to establish contact between the tool and the Frontend. The Frontend after parsing a command needs to be able to invoke Works Manager or tool functions. Tools or Tool Bearing Hosts need to be able to check file references or other resource requests with the Works Manager or use Frontend display or other services. Error conditions need to be reported and handled appropriately. Most of the above are services that will be implemented as collections of procedures that must be invoked by other often remote processes, be given arguments, and be able to produce return information. The mechanisms for invoking these services

need to handle synchronous and asynchronous control structures. 6c1c1

Detailed analysis of the above types of needs led us to view an appropriate solution for them as consisting of three main components: 6c1d

1) A Distributed Programming System (DPS) 6c1d1

The builders of the NSW as well as possible future builders of tools and services must create a system that spans both process boundaries in the same machine and process boundaries across different machines. Facilities must be provided at the level of easy system building access and in forms the systems builder is accustomed to dealing with, that specify primitives for: 6c1d1a

a) Creation and manipulation of various forms of distributed process structures 6c1d1a1

b) Establishing communication paths between processes 6c1d1a2

c) Allowing processes to access each others services and communicate with each other 6c1d1a3

d) Handling error conditions 6c1d1a4

e) Standardized data structure types 6c1d1a5

2) Appropriate lower level interprocess communication facilities 6c1d2

One goal here is to provide appropriate modularity so that dependence on a particular communications network or technique is well isolated and can be replaced if needed. Above this level one needs an appropriate Host-Host protocol; and above that level, appropriate primitives for packing service requests and replies, encoding data structure types, and handling error conditions. Efficiency of response and throughput are key design goals within the constraints of the particular network chosen. 6c1d2a

3) Standard interfaces to classes of operating system and host services 6c1d3

The variety of hosts and operating systems expected to make up the NSW require some set of standards to be established for classes of services required by the Works Manager and other NSW components. Without such standards NSW system components would need much host/operating system dependent code and communications formatting. Packages of standard services are needed in such areas as file manipulation and transmission, login and accounting, batch job entry and so forth. 6c1d3a

Because of the diverse operating systems likely to be in the NSW, implementation and use of specified protocols by other contractors with different and valuable points of view, the novelty of network-based resource sharing system development, and the need for operational experience, NSW protocol development is inherently an iterative evolutionary collaborative process. The initial version of NSW protocols is based on the concepts outlined below. Work proposed here will take the protocols through integration and shakedown, and then through review into the next design iteration, which should be close to what will exist in the final NSW. 6c1d4

PAST PROTOCOL DESIGN APPROACHES 6c2

Several ARPANET applications protocols have been designed and implemented since the Host-Host Protocol was adopted in 1970. Most have been bootstrapped from lower level applications protocols. For example, the File Transfer Protocol (FTP) was built upon Telnet, and the Remote Job Entry Protocol (RJE) upon both Telnet and FTP. The highest level protocol shared by all such bootstrapped protocols is Telnet. 6c2a

Although the bootstrapping principle is a sound basis for Network protocol development, we believe that Telnet, providing little more of use than a standard character set, is NOT the most appropriate foundation for a large class of applications protocols. 6c2b

Each application protocol has had to develop its own syntax for handling service requests and replies, its own error handling mechanisms, its own interprocess control conventions and so forth. The result is that each

applications protocol implementation has redundant special user and server processes to perform these functions when in fact they could be shared if a different design approach were taken. 6c2c

OUR APPROACH TO NSW PROTOCOL DESIGN 6c3

INTRODUCTION 6c3a

The approach we have taken to NSW protocol work consists of the three areas mentioned above: 6c3a1

1) Development of a Distributed Programming System on which higher level application protocols can be built and that can serve in its own right as the basis for communication and control of NSW to cooperating distributed processes. 6c3a1a

2) Development of appropriate higher level protocol packages needed by NSW. 6c3a1b

3) Study of Host-Host protocol needs. 6c3a1c

DISTRIBUTED PROGRAMMING SYSTEM (DPS) 6c3b

A DPS containing, among other features, a Procedure Call Protocol (PCP)--a Network-Standard mechanism for invoking arbitrary named, argument-driven, and result-producing procedures in a remote process--is a much more appropriate and powerful foundation for many applications protocols than the approach presently in use described above. 6c3b1

Such an approach: 6c3b2

1) Meets the NSW needs described above and in addition, 6c3b2a

2) Expedites the specification of applications protocols by permitting their documentation to have a functional, rather than a syntactic orientation. 6c3b2b

3) Largely eliminates the need for separate, application-specific user processes. 6c3b2c

(Present application protocols each require special user and server processes to encode and decode their communications.) 6c3b2c1

4) Reduces the cost of making large, existing software systems available as Network servers by allowing a Network interface more compatible with their internal organization. 6c3b2d

(i.e., most services are implemented as collections of primitives accessible by procedure calls.) 6c3b2d1

5) Provides the basis for a more natural interface between local and remote procedures, and therefore 6c3b2e

6) Encourages the sharing of software, by making procedures on remote hosts as accessible to the programmer as local ones. 6c3b2f

The DPS permits a collection of processes within one or more ARPANET hosts to communicate at the procedure call level. In effect, it makes component procedures of remote software systems as accessible to the programmer as those within his own system. 6c3b3

At the highest level, the initial DPS is the specification of a virtual programming environment in which remote procedures are assumed to operate. The model specifies the manner in which remote procedures gain and relinquish control, the kinds of data structures with which they can be expected to deal, and so forth. One of the tasks of the DPS implementer, therefore, is to provide a mapping between his real programming environment and the virtual one defined by DPS. 6c3b3a

At a slightly lower level, DPS is the specification of the interchanges between two connected processes which implement the virtual programming environment. 6c3b3b

The multi-process systems whose construction DPS makes practical, and of which the NSW is an example, consists of collections of "procedures" and "data stores" called "packages", in "processes", interconnected in a tree structure by "channels". Procedures within a process have free access to the procedures (and data stores) of each process adjacent to it in the tree structure, and may call upon them as if they were local subroutines. Superimposed upon the tree structure is a more general set of interconnections which give a non-adjacent processes in the tree the same kind of access to one another. 6c3b4

APPLICATION LEVEL PROTOCOLS AND TOOL ENCAPSULATION 6c3c

The various operating systems, executives, and other host facilities that are widely used by the NSW need to be appropriately packaged both to fit into the framework used for the Distributed Programming System and to provide a standard interface to the Works Manager and other NSW components that may need to use them. For example, each operating system provides a file system with different primitives for manipulating files, different data types or encodings of data types and so forth. To provide for manipulation of these different file systems and for file transfers, a standardized collection of file operations has been defined. Similar collections have been defined for other domains such as handling remote job entry. A tool encapsulation package has been defined for implementation on Tool-Bearing Hosts (TBH) to ensure that the Works Manager and other NSW needs such as proper error handling can maintain the NSW integrity. 6c3c1

HOST-HOST PROTOCOL LEVEL 6c3d

The initial NSW will make use of the Host-Host protocol presently in general ARPANET use. A new experimental Host-Host protocol has been under development at Stanford University, and PDP-11 ELF and PDP-10 Tenex implementations are being built. We would expect, as part of the continuation work herein proposed, to work with the implementers in setting up experiments to evaluate the relative merits of these two protocols for NSW ongoing use. Further, one can consider developing yet a third Host-Host protocol tailored for use with the DPS. 6c3d1

ACCOMPLISHMENTS 6c4

i) The protocol needs of the NSW were evaluated along with current ARPANET protocols and it was concluded that: 6c4a

a) The current Host-Host protocol with parameters appropriately set would be suitable for initial NSW use. 6c4a1

b) The basis for current ARPANET higher level application protocols was not on firm footing for NSW development and a new basis was needed. Such a basis was developed (documented, designed and implemented for PDP-10 and PDP-11 computers) in the form of the DPS described above. 6c4a2

c) Application service packages needed by the NSW were designed and documented. 6c4a3

2) Consultation on general ARPANET protocols was provided. 6c4b

3) Notes accurately recording the status of all ARPANET protocols have been maintained. A summary of this material giving current developments, annotated bibliographic information, and names of people active or knowledgeable in each area was published as RFC 661 NIC (31203) and widely distributed in the ARPANET community. 6c4c

4) Measurements and collaboration with BBN personnel and others to determine the reasons behind low throughput and response of the ARPANET and connected hosts was undertaken during March. 6c4d

The results of this work have led to improvements in the Tenex NCP and scheduler, clearer understanding of the need for ARPANET reconfiguration, and the need for changes in its basic communication algorithms and parameters. 6c4d1

5) Active participation was carried out with MCA and others at SRI involved with the NSW design relating protocol issues to overall NSW System design. 6c4e

SPECIFIC OBJECTIVES FOR THIS PROPOSAL 6c5

1) Achieve consensus with other NSW contractors on the set of NSW protocols required to control NSW resources and provide interprocess communication. 6c5a

2) Complete integration and shakedown of the DPS protocol on the PDP-10 for both communication between processes on the same Tenex, across Tenexes, and to other hosts. 6c5b

3) Complete integration and shakedown of the DPS protocol on the PDP-11 for communication between processes on ELF and hosts. 6c5c

4) Redesign and document the NSW protocols-based on the experience at SRI and of other NSW contractors, needs for error and crash recovery, and other needs specified by NSW contractors. 6c5d

5) Perform measurement and efficiency studies of NSW

protocol implementation in collaboration with other NSW contractors. 6c5e

PROPOSED PROTOCOL TASKS 6c6

The tasks necessary to meet the above objectives are the following. 6c6a

1) INTEGRATION AND SHAKEDOWN (4mm) 6c6b

Integration of the NSW system will not have been completed at the end of the current contract period. Once initial integration is achieved, continuing use will uncover bugs and problems that must be dealt with. There may be a need as part of this shakedown process to implement some ad-hoc design changes to deal with crash and error recovery early and without waiting for the next major implementation cycle to come in the follow-on contract. If more time than estimated above is required, some other task such as the measurement task may have to slip or be less extensive. 6c6b1

2) MEASUREMENTS AND EFFICIENCY (3mm) 6c6c

At the present time ARC has been obtaining remote computer support over the ARPANET from a TENEX system operated by Bolt Beranek and Newman (BBN). Responsiveness and throughput through the Tenex operating system and ARPANET are below what we consider one can reasonably expect. Therefore, we have had to divert some extra effort into creating diagnostic and measurement programs to help isolate the problem as there is no agency charged with an overall responsibility. Initial experience with the NSW in the ARPANET environment is likely to show similar problems requiring measurement and diagnostic programs. Areas requiring measurement and efficiency studies are described below: 6c6c1

Distributed Programming System and Application Protocols 6c6c2

There is a need to measure the performance of the communications aspects of the NSW system as implemented by the Distributed Programming System (DPS). These measurements should focus on the processing overhead in the encoding and decoding of the procedure calls and returns for transmission via the ARPANET standard Host-Host protocol, the data encoding actually transmitted, and number of ARPANET

messages required to perform frequently used NSW operations. As directed by these measurement results, efficiency improvements might require: 6c6c2a

Recoding indicated portions of the PCP mechanisms in the hosts. 6c6c2a1

Redefining the transmission encoding for PCP data structures. 6c6c2a2

Re-designing the Interprocess communications mechanisms used in Intra-host and/or Inter-host procedure calls. 6c6c2a3

Performing implementation changes based on available time and seriousness of problems. 6c6c2a4

Host-Host protocol Comparison Measurements 6c6c3

We plan to work with Stanford University and BBN in setting up measurements to compare the throughput and responsiveness of running the NSW communications using the existing Host-Host protocol and the new experimental protocol being developed at Stanford University. Based on the findings of these studies, appropriate recommendations will be made to the NSW steering committee. If the new protocol is found to have significant advantages, then other groups will have to be funded to implement it on non PDP-10 and PDP-11 hosts in the NSW. 6c6c3a

3) DPS VERSION 3 AND REVISED APPLICATION PACKAGES VERSION 3 (8mm) 6c6d

This task requires incorporating the lessons learned in the initial implementation--together with the suggestions received from other groups both within and outside the NSW community. We have recorded all suggestions made to us for improvements. This task will be a careful design of revised protocols taking into account all the lessons and suggestions made up to this point. Further, this redesign will incorporate capabilities to handle new needs such as for error and crash recovery, multiple Works Managers, and so forth. This design should be reviewed by other groups before implementation is begun. 6c6d1

4) TOOL BEARING HOST AND COLLABORATION CONSULTATION (7mm) 6c6e

Besides PDP-10 Tenex and PDP-11 ELF, other Tool Bearing Hosts to be brought into the NSW during the proposed contract period will be Honeywell 6180 Multics at MIT and IBM 360/91 at UCLA. 6c6e1

We will provide consultation for the installation of these systems as NSW Tool Bearing Hosts and for installation of tools residing on these systems. 6c6e2

This task will also provide the leadership on an NSW Protocol Working Group consisting of representatives from other NSW contractors with protocol interests to achieve consensus on interim and longer range NSW protocol designs. 6c6e3

5) DOCUMENTATION (4mm) 6c6f

Documentation will be of four types: 6c6f1

1) Documentation of the protocols actually implemented (2mm) 6c6f1a

2) Design documentation for the next iteration of protocol work (covered under design task above) 6c6f1b

3) Collaboration with other NSW contractors in the creation of a draft handbook for Tool Bearing Host suppliers (1mm) 6c6f1c

4) Final reports and study report documents (1mm) 6c6f1d

DELIVERABLES 6c7

Deliverables will include the documentation described above, source and object code files for DPS implementations for the PDP-10 Tenex and PDP-11 ELF systems, reports on findings of measurement and efficiency, and crash and error recovery studies. Consulting as described will also be delivered. 6c7a

III TOOLS 6d

Assuming the successful operation of the basic NSW system framework, the ultimate success of the NSW will depend on the usefulness and power of the tools available to users. ARC is vitally interested in seeing that powerful tools for important system building and support tasks are provided to the NSW for document preparation, and publication, program preparation, debugging and maintenance, message and dialog support, and other areas. 6d1

SPECIFIC OBJECTIVES FOR THIS PROPOSAL 6d2

1) Install the NLS Base, Calculator, Sendmail, User Options, User Subsystems, Output Processor, Graphics capabilities (NLS-8 plus capabilities) as an NSW tool or tools, using if possible existing WM services. 6d2a

2) Work with MCA to specify an identification system for NSW with needed services for NLS. 6d2b

3) Install as an enhancement to the NLS Base subsystem graphics capabilities for drawing flow charts, organization charts and other simple line drawings using a Tektronix 4014 display and work with NSW application users to refine as needed the user interface. 6d2c

4) Provide the following publication facilities. 6d2d

Modifications to the NLS Output Processor to support single page formatting, proofing of mixed text and graphics pages on the Tektronix 4014, and output to a COM device of formatted documents containing mixed text and graphics that meets Air Force documentation format standards. The COM service bureau we are currently working with can produce microfiche at 24 times reduction. A new lens would be required for 48 x fiche. Use of a different service bureau would require that it obtain or produce the required Singer 6000 software. We assume our current service bureau will produce the Singer 6000 software to support graphics as well as text.

If time permits, a new NLS heading entity with appropriate capabilities in the Output Processor and in NLS View Specifications. 6d2d1

5) Design and build a prototype workstation to support a graphics terminal and two-dimensional use with a graphical pointing device from a class of commercially available alpha-numeric terminals using the Digital Equipment Corporation LSI-11 family of modules. Additional support will be required in the follow-on contract to produce production specifications. 6d2e

6) Provide one full-time-equivalent person for prime work, resident at Gunter AFS, Alabama to: 6d2f

a) Help formulate plans for the use of NLS in building and documenting Air Force COBOL and other programs. 6d2f1

b) provide special documentation to aid Gunter personnel in the above work. 6d2f2

The exact design for how to carry out this role will require collaboration and mutual agreement with the appropriate Gunter personnel and NSW Steering Committee. A suggested design will be presented to these parties by two weeks after start of work on the tasks resulting from this proposal. 6d2f3

Note:

This task can only be carried out effectively if proper support is provided to this role by the Air Force in the form of adequate communication, terminal, and training facilities; adequate administrative and other support; and a person or persons who are charged with making this arrangement work and have the authority to provide resources and assistance as needed, and who are familiar with the Gunter application areas to be assisted. 6d2f4

ACCOMPLISHMENTS 6d3

Accomplishments in the tool area up to July 17, 1975 are or will be the following: 6d3a

1) NLS-8 was made operational at Office-1. 6d3a1

This task involved evaluation and modification of the NLS-8 user features, completion of the HELP database of online documentation, and shakedown and testing of the new system. NLS-8 is presently being used by NSW users and forms the basis for all future NLS development. 6d3a1a

2) Design and implementation of a new NLS file system to support graphics and other new data types. 6d3a2

3) Design and implementation of a graphics capability to allow documentation writers to create flow charts, organization diagrams, block diagrams etc. intermixed with their text. 6d3a3

4) Modifications to the Output Processor to support the Singer 6000 COM device. 6d3a4

5) Design and implementation of the modifications to the basic NLS system structure to allow it to fit fully into the NSW environment and utilize the services of NSW Frontend, Protocols, and Works Manager. 6d3a5

Testing and debugging of this work with the other NSW components will not be completed by the end of the current contract period. 6d3a5a

6) Creation of new subsystems to aid document production in areas of index and table of contents creation, formatting of documents in Air Force and other formats, and aids for editing the most frequent classes of transcription errors. 6d3a6

7) Development of new tab features to simplify table creating. 6d3a7

8) Interface of NLS files to appropriate output formats for the B4700 COBOL compiler. 6d3a8

9) proper handling on input of sequential files to NLS file structure conversion for selected Air Force document formats. 6d3a9

10) Changes to the HELP system to make it a generally available NSW facility for other tools as well as the NSW Frontend and Works Manager. 6d3a10

11) A letter subsystem to simplify preparation of network and hardcopy messages and letters for NSW clerical users. 6d3a11

12) Help in the form of programming to convert IBM 360/370 tapes to ASCII for input into NLS, consultation on bugs in Air Force 360 programs associated with this process, and consulting on data entry and formatting problems. 6d3a12

13) Extensive user documentation both online and in hardcopy, for NLS 8, the NSW Frontend, and new NLS features such as graphics for the NSW. 6d3a13

SPECIFIC TOOL TASKS 6d4

1) INTEGRATION OF THE NLS BASE SUBSYSTEM INTO THE NSW ENVIRONMENT (2mm) 6d4a

As described in the previous proposal, NLS has been modified to take full advantage of the NSW environment = Frontend, protocols, and Works Manager. This was done to provide a test bed for many NSW concepts for tools specifically developed or modified for the NSW environment and to provide the economies and responsiveness offered by the NSW Frontend. 6d4a1

Further, NLS, as a large and sophisticated set of interactive capabilities, provides a good "testbed" with which to see whether basic NSW components such as the Works Manager, Tool Encapsulation, and Frontend provide a complete set of services for Tools. 6d4a2

Work will not have been completed on the NSW system integration at the close of the current contract. The first NLS subsystem to be integrated will be the Base Subsystem providing editing, file, and other services. 6d4a3

2) COMPLETE THE INTEGRATION OF ADDITIONAL NLS SUBSYSTEMS AND SHAKEDOWN OF ALL OF THEM IN THE NSW ENVIRONMENT (9mm) 6d4b

When the previous task is completed, the remaining NLS subsystems need to be integrated and the total set of NLS capabilities shaken down in the NSW environment. What is involved here is the following: 6d4b1

For each of the subsystems - Programs, Calculator, Sendmail, User Options, Output Processor - a grammar database needs to be created. These subsystems need new initialization routines, changes to interface routines and procedure dispatches to interface with NSW protocols, and any changes necessary to utilize NSW Frontend services.

User subsystems such as Format, Modify, and Publish will be combined into a new system level subsystem for support of documentation. Other User subsystems and

Content analyzer programs will require additional funding for their conversion to this new NLS system. 6d4b1a

To install the Sendmail Subsystem will require the MCA provided NSW identification system mentioned above. Further, there are several alternative strategies, in installing this tool for how interface is made to the NSW file system and Works Manager services. Some additions or changes in Works Manager services are expected to be required and thus Sendmail installation strategy would be worked out with the mutual agreement of MCA, the TCC, and NSW Program Management Officer. 6d4b1b

3) GRAPHICS IN THE USER ENVIRONMENT (3mm) 6d4c

The graphics equivalent of the "editor" is new and is likely to undergo evolution as users gain experience in its use in real applications. The command set, of a prototypical model, controls the creation and management of graphical figures and textual material. The interface may not be as natural as it can and should be for actual Air Force applications. Drawings are constructed in a data structure which is analogous to the acetate cells used by animators and illustrators. Figures can be drawn onto the cell, and a number of cells can be overlayed on one another to create a whole illustration. These named cells can be moved relative to each other and can have any number of subcells fastened to them. A cell can be used as a template to create a copy of a common figure possibly transformed in size, position, and rotation. As users gain experience with the system, commands will be deleted from, and added to the repertoire. User interface facilities for pointing and selection will be improved to more closely match the evolving NSW environment. 6d4c1

4) DOCUMENTATION PRODUCTION TASKS (9mm) 6d4d

The following tasks need to be performed to provide a self contained documentation production environment for presently known Air Force needs: 6d4d1

Modification to the Output Processor to make it a page oriented formatter. 6d4d2

The present Output Processor requires one to treat a document as a single file, and any changes to the

format of a single page requires passing through the entire document. 6d4d2a

For large Air Force manuals that must reside online in multiple files and undergo maintenance on a page at a time basis, these restrictions are both awkward and expensive. We propose making those necessary modifications to remove these problems so that pages can be maintained and formatted individually. This task is also a prerequisite for the next task. 6d4d2b

Allow the Tektronix 4014 display unit to be used to display formatted page proofs for material to be sent to COM. 6d4d3

This task would allow formatting of mixed text and graphics on a single page-at-a-time basis for display on the Tektronix 4014 display and for final COM output. 6d4d3a

Provide a Heading Entity. 6d4d4

The new user feature to provide a new NLS entity for headings could not be completed as we had hoped by July. It would have allowed NLS viewspecifications and Output Processor directives to take full advantage of it for portrayal online and in hardcopy. If time permits during this proposed extension we still believe it is important and should be implemented. 6d4d4a

To accomplish output to COM a specially formatted tape is prepared by the Output Processor. This tape is then input to the COM unit. Special software must reside on the COM unit to transform the OP-provided format to its internal instruction set. We are not proposing to provide this software. Our current COM suppliers of Singer 6000 services and III Comp-80 services have provided this software. They will have to make additions to support graphics as well. Conversations we have had with them indicate they will make these additions. 6d4d5

5) LSI-11 TERMINAL CONTROLLER(4mm) 6d4e

NLS is designed to be used from both typewriter and two-dimensional displays. To provide low cost two-dimensional displays for use with NLS, a special terminal controller we call a Line Processor (LP) has been constructed from a 4 bit microprocessor. This

device initially controlled a graphical pointing device (Mouse), a special keyboard called a keyset, and a class of commercially available low-cost alpha-numeric displays to achieve a true two-dimensional text workstation [36]. Since its original development the LP has been used to control magnetic tape cassette units and a Tektronix graphics display. 6d4e1

While this Line Processor configuration provides adequate service for simple alpha-numeric applications, it is inadequate to provide for: 6d4e2

a) Alpha-numeric applications such as screen sharing 6d4e2a

b) graphics applications such as cursor increment control and the refreshed (non-storing) display of lines and text on the graphics display. 6d4e2b

In addition to these limitations, the organization of the existing Line Processor program (read only memory based) makes remote maintenance difficult and costly. 6d4e3

The 4-bit microprocessor utilized in the existing unit was the first development of a rapidly expanding technology. The LSI-11 now provides a processor which is virtually identical to the PDP-11/40 utilized for the Frontend machine at a price comparable to the current LP. The increased power of the LSI-11 will allow the LP to handle the screen sharing and graphics applications precluded by the use of the 4-bit microprocessor. 6d4e4

More importantly, the maintainability of the LP will be greatly enhanced by: 6d4e5

a) Utilization of a high level language for implementation of LP software (L10). 6d4e5a

b) Utilization of a read/write memory for program storage. This approach allows the user to obtain the most current (or appropriate) version of LP software, and to obtain special software for hardware checkout, communication line checking, and special applications (such as a calculator). 6d4e5b

The reserve CPU power of the LSI-11, in this application, opens the door to improved work station performance through the use of local mass storage to provide a stand alone capacity impossible with the existing unit. 6d4e6

During this contract period manpower will be provided to build and test only the initial prototype LSI-11 Line Processor. Follow-on funding will be needed to provide for the manufacture and support of field units. Advanced applications such as local mass storage and specialized workstation software (for example, the calculator) are also not provided in the 4mm figure. 6d4e7

6) TOOL DOCUMENTATION (6mm) 6d4f

Tool documentation will consist of the following tasks: 6d4f1

a) System documentation 6d4f1a

Commented source code listings will be provided. 6d4f1a1

b) Hardcopy user documentation 6d4f1b

Hardcopy documentation such as command summaries, cue cards, and others will be provided on a best effort basis. 6d4f1b1

c) Online user documentation 6d4f1c

Help database descriptions of new tools and services will be provided. 6d4f1c1

7) CONSULTING WITH NSW NLS USERS AT THE AF DATA SYSTEM DESIGN CENTER (9mm) 6d4g

NLS will be a key tool for use by programmers, managers, and system documenters at the DSDC. There will be a need to achieve the following objectives for proper technology transfer: 6d4g1

a) Assist DSDC personnel in the analysis of their tasks and in the development of methodology to make maximum use of NLS in these tasks. 6d4g1a

b) Provide general NLS training as required. 6d4g1b

c) Develop scenarios or other special documentation, as time permits, oriented toward DSDC use of NLS in its tasks. 6d4g1c

d) Assist one or two DSDC staff members in achieving a capability in the use of NLS such that these DSDC

staff can carry on in the future with analysis, training, and other tasks associated with use of NLS. 6d4g1d

e) Assist the above DSDC staff in planning for this type of activity beyond this contract period. 6d4g1e

The detailed design of how to achieve the above objectives needs to be worked out in collaboration with DSDC personnel and the NSW Steering Committee. The following general approach can form a starting point for that design: 6d4g2

a) Resources in the form of terminals, communications, and space for training and offices for SRI personnel need to be available. 6d4g2a

b) Appropriate high level DSDC management support for this effort is required. 6d4g2b

c) A single focal point person from the DSDC staff is needed for SRI interface. Preferably this person (or small number of people) will have been given enough time for this assignment to become an expert in NLS usage and can play a strong role in adapting this new technology to DSDC needs with SRI assistance. 6d4g2c

It is our experience that technology can only be successfully transferred if there is a person(s) within the target organization very knowledgable about both his organization's needs and the potential of the new technology, and who has the time and interest to follow through on the transfer. 6d4g2c1

d) No single ARC person has the detailed background in all the expected DSDC application areas, programming, documentation, and management. Therefore, we would propose assigning one person skilled in general NLS use to provide continuity and follow-on support to work with SRI specialists in the application areas and DSDC personnel to achieve this task's objectives. 6d4g2d

The specialists would spend appropriate periods at DSDC for assistance and task analysis and planning of NLS use. We would expect that the continuity person would spend roughly half the contract period at DSDC overlapped with the specialists and beyond

for follow through. This person would also be responsible for scenarios or other special documentation. 6d4g2d1

INTERDEPENDENCIES WITH OTHER NSW CONTRACTORS: 7

The purpose of this section is to list task areas requiring joint work with other NSW contractors to aid ARPA in coordination of proposals. 7a

FRONTEND 7b

The tasks involving close collaboration with other NSW contractors are: 7b1

Error and Crash recovery (MCA, BBN and other TBH suppliers) 7b2

New Terminal Types (ADR) 7b3

ELF Operating System (ADR) 7b4

control Meta Language (Tool suppliers and MCA) 7b5

PROTOCOLS 7c

All work in this domain requires interaction, consensus, and review with MCA and others on design. BBN and other TBH suppliers must implement various levels of protocols and be prepared to assist in measurement and possible performance tuning work. 7c1

TOOLS 7d

MCA needs to provide an NSW identification system that meets NLS needs such as Sendmail. 7d1

USER DOCUMENTATION 7e

General standards for NSW user documentation must be negotiated among ARC, MCA, and other contractors offering NSW services. Cooperation must take place in writing about certain areas, such as the Frontend and Works Manager, where more than one contractor affects the appearance of a tool to the user. 7e1

STATEMENT OF WORK: 8

INTRODUCTION 8a

The tasks listed below will be accomplished within an environment having the following areas of uncertainty: 8a1

1) We expect the responsiveness and reliability of PDP-10 Tenex computer capacity available to ARC over the ARPANET after July 1, to be improved over what we have experienced since March 1. The time it takes to perform online tasks since March appears to have increased over what it was previously when we had our own local machine as a result of decreased responsiveness. The machine we will be using will be at USC-ISI, and at this time we have no experience using it in a fully loaded condition. 8a2

2) Integration of the various NSW pieces could turn up unexpected problems in the base technology on which the NSW is built--computer hardware, operating systems, ARPANET, and Host-Host protocol. 8a3

3) Integration of the various NSW pieces could turn up unexpected gaps in expectations or understandings at the interfaces between the various NSW contractors. 8a4

4) More time than anticipated could be required to deal with communications and education between ARC and NSW management, NSW users, and other contractors. 8a5

5) There could be shifts from our present understanding of priorities and requirements as reflected in this proposal. Given the new high technology environment in which the NSW must operate and the newness of this technology and its possibilities for user groups, it is not unreasonable that shifts may occur. 8a6

6) Other NSW contractors, we assume, have funding and work statements for tasks in which we will require their collaboration, such as measurement and error and crash recovery (an area minimally funded during the proposed period and of some concern to us). 8a7

If, as a result of any of the above or other factors, changes in work statement or direction are required, the NSW Steering Committee will be informed and its guidance sought. Tasks which, while important, we feel could slip to provide possible slack are identified below. 8a8

Within two weeks after the start of work resulting from this proposal, detailed milestones will be provided to the NSW Technical Coordination Contractor (TCC). Progress reports will be provided the TCC monthly. With the milestones, existing design documents will be provided the TCC. The TCC will be notified of changes in design as these are found to be required. 8a9

FRONTEND (FE) TASKS 8b

1) The Works Manager (WM) user interface will be integrated with the WM on the same and different PDP-10 Tenex systems. This task requires delivery of a complete and debugged (as much as this is possible) Command Language Interpreter (CLI) and Control Meta Language (CML) and L-10 compilers. CML and L-10 compilers will run as NSW tools on a PDP-Tenex with an NLS source language environment. Necessary implementations of NSW protocols to support FE WM intercommunication are also required for this task. 8b1

2) It will be a goal to support 20 users simultaneously on a PDP-11 Frontend. This will require use of disk storage as the PDP-11 does not have adequate main memory capacity to support that number of users. ARC will assume responsibility for refining the release of virtual memory (VM) ELF available from Speech Communications Research Laboratory (SCRL) into an operating system which can support the NSW FE software. The version of VM ELF containing all the needed features for the NSW FE is not scheduled for release by SCRL until August. This release will allow loading from the network of FE software as a user program. Earlier releases may prove useful and will be used as appropriate. ARC will accept responsibility for making operational code already written by SCRL that utilizes disk storage for virtual memory. This task is one that could be slipped at the cost of running with fewer users for a period of time, although error and crash recovery analysis may show that use of disk storage is required for storage of needed redundant information. The DPS protocols will be implemented running under ELF. A Command Language Interpreter will run under ELF as a user program and communicate appropriately with WM and NSW tools. The CML compiler and associated part processor will produce grammar data structures for the PDP-11 version of the FE. An L-10 cross compiler and run time environment will exist from the PDP-10 to PDP-11. A cross network debugging environment from PDP-10 to PDP-10 or PDP-11 for FE maintenance will be produced. 8b2

3) Three terminal classes will be supported by the FE:

half-duplex line-at-a-time typewriter terminals, full duplex character-at-a-time typewriter terminals and two-dimensional CRTs using the ARC developed Line Processor. ARC will write a device driver for the PDP-11 to support Line Processor based workstations. All three classes will be supported on the PDP-10 through a TIP. Half-duplex line-at-a-time terminals will not be supported on the PDP-11 within the funding proposed here. If this class of terminals is to be supported from the PDP-11, additional funding will be required. 8b3

4) We will collaborate with other contractors in crash and error recovery analysis and design of appropriate strategies. Some implementation changes may be required during this contract period. Depending on level of effort required here, slippage of other tasks could result as only minimal funding is included in this proposal for such implementation work. The NSW Steering Committee's guidance will be sought as appropriate. 8b4

PROTOCOLS 8c

1) ARC will provide one full-time equivalent person (one person with prime responsibility with assistance from others as required) to serve as chairman of a working group to finalize NSW protocols. By scheduling meetings, making personal visits to other contractor sites (the sites and their responsibilities are: SRI-Frontend and NLS, MCA-works Manager, BBN-Tenex, MIT-Multics, UCLA-360), this person will provide consulting on protocols and tool installation and work to achieve a consensus on the set of NSW protocols required to control NSW resources and provide appropriate communication. Meetings will be held by the end of July to achieve an interim consensus for initial 360 and Multics implementations with more thorough review in November after initial implementation, usage and measurement experience, and updated documentation exist. Consensus and redesign of a Version 3 family of protocols will be achieved and documented by the end of the contract period. 8c1

work on crash and error recovery or unexpected integration problems may require slippage of the measurement work. There is no explicit funding in this proposal for implementation of changes required to support crash and error recovery. Major changes required in this area will show up in Version 3, although some interim changes may be required during this contract period. Version 3 protocols will not be implemented this contract period. Changes in DPS arrived at in the interim consensus will be implemented for PDP-10 and PDP-11. We assume any interim changes will

be minor and will include only those necessary to get an initial Multics and 360/91 NSW protocol implementation operational. Any extensive changes felt desirable will be reflected in the Version 3 redesign. Application level package implementations will be the responsibility of other contractors. We assume that the other contractors listed above will participate in measurement activities. 8c1a

TOOLS 8d

1) Install the NLS Base, Calculator, User Options, User Subsystem, Output Processor, and Graphics capabilities (NLS-8 plus capabilities)-as an NSW tool or tools, using, if possible, existing WM services. 8d1

2) Work with MCA to specify an identification system for NSW with needed services for NLS. 8d2

3) Install, as an enhancement to the NLS Base subsystem, graphics capabilities for drawing flow charts, organization charts and other simple line drawings using a Tektronix 4014 display and work with NSW application users to refine as needed the user interface. Specifications for the initial set of capabilities to be provided are described in Attachment X. Those specifications are not to be part of this contract, but the NSW Steering Committee will be informed of any changes. 8d3

4) Provide the following publication facilities: 8d4

Modifications to the NLS Output Processor to support single page formatting, proofing of mixed text and graphics pages on the Tektronix 4014, and output to a COM device of formatted documents containing mixed text and graphics that meets Air Force documentation format standards. The COM service bureau we are currently working with can produce microfiche at 24 times reduction. A new lens would be required for 48 x fiche. Use of a different service bureau would require that it obtain or produce the required Singer 6000 software. We assume our current service bureau will produce the Singer 6000 software to support graphics as well as text.

If time permits, a new NLS heading entity with appropriate capabilities in the Output Processor and in NLS View Specifications. 8d4a

5) Design and build a prototype workstation to support a graphics terminal and two-dimensional use with a graphical

pointing device from a class of commercially available alpha-numeric terminals using the Digital Equipment Corporation LSI-11 family of modules. Additional support will be required in the follow-on contract to produce production specifications. 8d5

6) Provide one full-time-equivalent person for prime work, resident at Gunter AFS, Alabama to: 8d6

a) Help formulate plans for the use of NLS in building and documenting Air Force COBOL and other programs 8d6a

b) Provide special documentation to aid Gunter personnel in the above work. 8d6b

This task can only be carried out effectively if proper support is provided to this role by the Air Force in the form of adequate communication, terminal, and training facilities; adequate administrative and other support; a person or persons who are charged with making this arrangement work and have the authority to provide resources and assistance as needed, and who are familiar with the Gunter application areas to be assisted. 8d7

The exact design for how to carry out this role will require collaboration and mutual agreement with the appropriate Gunter personnel and NSW Steering Committee. A suggested design will be presented to these parties by two weeks after start of work on the tasks resulting from this proposal. 8d8

DOCUMENTATION 8e

1) Commented source code for all software produced. 8e1

2) User documentation in the form of an online Help file, cue card, and command summary for all user features specified above. 8e2

3) System overview description of the SRI NSW FE, including its interfaces to ELF and protocols. 8e3

4) Draft version of an NSW Tool suppliers manual describing use of FE services. 8e4

5) Draft material to be integrated with additional material to be supplied by MCA to specify what software must be implemented for a computer not currently on the ARPANET to make it an NSW Tool Bearing Host. 8e5

REFERENCES: 9

(1) Robert Balzer (USC-ISI), T.E. Cheatham (HARV-10), Stephen Crocker (ARPA-IPT), Stephen Warshall (MCA), Design of a National Software Works. USC/Information Science Institute, Marina del Rey, California. ISI-RR-73-16. November 1973. 25p. 19208. 9a

(2) Charles H. Irby. "Display Techniques for Interactive Text Manipulation". In: Proceedings of National Computer Conference, 1974, p.911-919. 20183. 9b

(3) William E. Carlson (AFDSC-AFDAA), Stephen D. Crocker (ARPA-IPT), "The Impact Of Networks On The Software Marketplace." 24309. 9c

(4) Jack Goldberg (Editor), Proceedings of the Tri-Service Symposium on the High Cost of Software, Held in Monterey, California, Stanford Research Institute, September 17-19, 1973. 9d

(5) SRI-ARC PROPOSAL FOR RESEARCH. ISU 74-132, Development Support NSW. 23352. 9e

(6) Donald I. Andrews. "Line Processor:-A Device for Amplification of Display Terminal Capabilities for Text Manipulation". In: Proceedings of the National Computer Conference, 1974, p.479-487. 20184. 9f

(7) Charles H. Irby, Charles F. Dornbush, Kenneth E. Victor, and Donald C. Wallace, Control Meta Language. Chapter in "Augmented Knowledge Workshop". To be published by SRI, June 1975. 22133. 9g

(8) M.C. Grignetti et al An Intelligent Online Assistant and Tutor-NLS Scholar. In: AFIPS Conference Proceedings, Vol. 44, Anaheim, California, May 1975. 25054. 9h

(9) Jonathan B. Postel. "National Software Works Protocols". 1 January 1975. SRI-ARC document 24856. 9i

(10) James E. White. "The Procedure Call Protocol". 1 January 1975. 24855. 9j

APPENDIX A NSW DEBUGGING ENVIRONMENT: 10

OVERVIEW - SOME DEBUGGING APPROACHES 10a

Over the years there have been a number of debugging approaches used to aid the development and maintainance of programs. Each approach has its relative merits and drawbacks. 10a1

CONSOLE DEBUGGING 10a2

In this approach a programmer uses the operator's console to examine and modify physical core locations. 10a2a

The nice features of this approach are: 10a2b

This method usually does not require that there be separate versions of a program for debugging and for production use of the program. 10a2b1

It is frequently possible to select a specific memory location and to cause the CPU to halt if the selected location is read and/or written and/or contains an instruction that is to be executed. 10a2b2

It is usually possible to execute a program one instruction at a time, thus making it possible to examine the state of the program any time it is desired. 10a2b3

There are many drawbacks to this approach: 10a2c

This approach is totally unsuited for debugging programs, other than the monitor itself, that are running in a multiprogramming environment. 10a2c1

This approach is difficult to use on a system that supports either a virtual or paged memory system. 10a2c2

Only one person can be debugging at a time. 10a2c3

The medium of information exchange is restricted to console lights and switches. 10a2c4

This is often a very tedious and unforgiving (of operator mistakes in keying in data for example) approach. 10a2c5

CORE DUMPS 10a3

In this approach, a programmer receives (usually after some

catastrophic condition has occurred, but sometimes in response to a programmed request) a core dump. The core dumps run the gamut from an octal dump of the entire contents of physical memory to highly formatted symbolic dumps of selected areas of a virtual address space. 10a3a

The advantages of this method of debugging are: 10a3b

This method usually does not require that there be separate versions of a program for debugging and for production use of the program. 10a3b1

The entire state, at the time the dump is made, is available. 10a3b2

The disadvantages of this approach are: 10a3c

It usually provides after-the-fact information about a program that has gone bad. Frequently the cause of the error occurred before the state got bad enough to cause a dump to be made. 10a3c1

When dumps are only available after some catastophic condition has been reached, a programmer gets no feel of the flow that the program took to get to this state. 10a3c1a

Frequently, much more information is provided than is desired. 10a3c2

In the case of numeric memory dumps, it is difficult to glean the desired information. 10a3c3

In the case of requested (selective) dumps, this method requires separate versions of the program for debugging and production use. 10a3c4

This method is infamous for wasting huge amounts of line printer paper. 10a3c5

PROGRAM TRACES 10a4

In this approach the programmer puts together a version of a program with debugging checks as part of the program. These checks take the form of decision point traces, procedure call traces, etc. The resulting traces can either be presented to the user on some hardcopy device, or a history

file can be produced that can then be analyzed by perhaps some other debugging aid. 10a4a

The advantage of this approach is: 10a4b

It can provide the programmer with a good picture of the flow control of the target program. 10a4b1

The drawbacks associated with this method are: 10a4c

The programmer, while he may be able to tell when a program has gone awry, is usually not able to modify the program and to try different control paths. 10a4c1

Occasionally a program will function properly with the debugging checks in place, and will function differently (improperly) without the debugging checks. 10a4c2

This approach may provide much more information than is desired. (It may be quite a while before a program goes bad and much useless information may have been collected in the meantime.) 10a4c3

INTERACTIVE DEBUGGING 10a5

In this approach a programmer monitors the program as it is running and has the ability to set (conditional) breakpoints, examine, and change his program to try different tacks. This approach, depending on the complexity of the debugger, can provide anywhere from octal debugging to high level language debugging. 10a5a

The advantages of this approach are: 10a5b

It does not require separate production and debugging versions of a program. 10a5b1

A programmer can set a breakpoint at the beginning of a suspect code and then step through the program (at any desired rate) to see where it goes bad. 10a5b2

Suspect code and/or state information can be modified and then retried. 10a5b3

A sophisticated debugger can provide the programmer with the ability to examine and modify his program using the source language that the program is written in. 10a5b4

The drawbacks of this approach are: 10a5c

This approach requires intuition on the programmer's part to get to the appropriate area for breakpoint setting but then intuition and hypothesis formation are requirements for debugging generally. 10a5c1

OBJECTIVES 10b

Since the environment to be provided by NSW even for dealing with setting up and handling batch jobs is an online interactive one, and since interactive debuggers are generally accepted to be the most powerful, we are proposing interactive debugging tools for the development and maintenance of the NSW, NSW tools, and programs developed within the NSW. 10b1

As mentioned briefly above, interactive debuggers can provide a wide spectrum of capabilities. Thus it is necessary for us to decide which capabilities are desirable, and which capabilities are feasible to implement, both during the period of this proposal and in succeeding years. One of the more important points to keep in mind is the goal of providing a "consistent" interface to the user, regardless of the program (and procedural languages it is written in) that is being debugged and regardless of the sophistication of the debugger (this means providing upwards compatibility as the debugger environment grows in sophistication). 10b2

DESIRABLE CAPABILITIES 10b3

The interaction techniques for communication between the user and the debugger should be concise, easy to remember, and invariant (as much as possible) across different target languages so that the programmer can concentrate his effort on debugging the target program and not on the mechanics of debugging. 10b3a

A good debugger should have the ability to examine any single or group of locations in the logical address space of the program being debugged. 10b3b

The debugger should provide several different forms for presenting location contents for examination to the user: 10b3b1

Numerically - in any base requested by the user. 10b3b1a

Low-level symbolically - in terms of the assembly

language of the physical machine that the program is running on. 10b3b1b

Full symbolically - in terms of the high-level language that the program was written in; this includes a knowledge of the data types supported by a language so that, e.g., data could be presented as L10 records. 10b3b1c

Textually - as a text string. 10b3b1d

Others - some other presentation forms may include presenting a text string as a sequence of numeric bytes, etc. 10b3b1e

A good debugger should have the ability to modify any single location or group of locations in the logical address space of the program being debugged. This should include the ability to "insert" a sequence of instructions, i.e. a patch, where previously there was only a single instruction. 10b3c

The debugger should provide several different forms for the user to specify the new contents of the target locations: 10b3c1

Numerically - in any base specified by the user. 10b3c1a

Low-level symbolically - in terms of the assembly language of the physical machine that the program is running on. 10b3c1b

Full symbolically - in terms of the high-level language that the program was written in. (Ultimately and ideally this implies the implementation of incremental compilers or a marriage between the debugger and some language interpreters or some compromise similar to the Compile Procedure capability of NLS-8.) 10b3c1c

Others - some other specification forms may include specifying a text string as a sequence of numeric bytes, etc. 10b3c1d

A good debugger should have a powerful breakpoint facility. 10b3d

This facility should include the ability to specify conditional breakpoints of perhaps the following two types: 10b3d1

A set of canned, frequently used conditions that the debugger knows about. 10b3d1a

An ability for the programmmer to specify some code that gets executed to determine whether or not the breakpoint should be taken. 10b3d1b

Once a breakpoint has been taken, it should be possible to have some prespecified actions take place. This might take on any of the following forms: 10b3d2

A programmer specifies that he wishes to see the value of one or more variables (program or state) upon encountering the breakpoint. 10b3d2a

A programmer specifies a sequence of debugger commands to be executed when the breakpoint is taken. 10b3d2b

A programmer specifies a sequence of code to be executed when a breakpoint is taken. 10b3d2c

It should be possible to set a breakpoint anyplace in the program. This includes setting a breakpoint to be taken when a data (as opposed to instruction) location is referenced. 10b3d3

In addition to a powerful breakpoint facility, a good debugger should have the following controls over the execution of the program being debugged: 10b3e

The ability to start the execution of the target program at any arbitrary location. 10b3e1

The ability to single step through a program. 10b3e2

The ability to execute a program in a (slowed down) trace mode. 10b3e3

The ability to interrupt a running and/or looping program and enter the debugger. 10b3e4

A good debugger should be able to execute "out-of-line" any arbitrary instruction that is specified either in machine language or in some higher level language. This includes the ability to do procedure calls from the debugger. 10b3f

A good debugger should be able to examine and modify the

state of the target program, e.g. examine and modifying the program call stack, etc. 10b3g

When a programmer examines and/or modifies the program state, ideally he wants to do this in terms of the language that the target program is written in and not in terms of the machine language of the machine that the program is running on. 10b3g1

A good debugger should provide sufficient capabilities so that a programmer can tell where he is: 10b3h

Did I get into the debugger via a breakpoint? If so, which one? If not, which instruction was interrupted to get me into the debugger? etc. 10b3h1

A good debugger should provide the ability for a user to get a typescript of his debugging session so that it can be examined at a later time or perhaps used as a training device for other programmers. In addition to being able to get a typescript, it should be possible to temporarily divert output to files other than the terminal the user is at. 10b3i

An ideal debugger would provide facilities for automatically updating source code files to reflect the patches made during a debugging session. 10b3j

A good debugger should be able to take advantage of the full power of the terminal that a user is currently using. 10b3k

For example, if a user is at a display screen, it might be nice to display several instructions on either side of the currently executing instruction and to have some indication of program flow presented graphically to the user. 10b3k1

A good debugger must be able to deal with any fork or process structure that is supported by the combination of the implementation language of the target program and the operating system under which the target program is running. 10b3l

SOME OBSERVATIONS 10b4

Upon examining the set of desirable capabilities, we see that a debugger can be roughly divided into the following two areas: 10b4a

Those areas of the debugger that are language independent. 10b4a1

Those areas of the debugger that are language dependent. 10b4a2

The borderline between these areas is in fact quite fuzzy. For example, the debugger command to examine a specific location is a facility that is desirable regardless of the implementation language of the target program; however, the format used to display the location contents to the programmer should be a function of the implentation language. Similarly, the debugger commands to show the top frame of the call stack or to call a procedure are valuable regardless of the target program's implementation language, but the format used to display the top frame and the code needed to implement the procedure call are very dependent on the implementation language. 10b4b

On the other hand, a debugger command such as "Show Data Record" while meaningful for languages such as PL/1 or L10 do not make any sense for a language like FORTRAN which does not support records. Similarly, data specifications are most likely implementation language specific. 10b4c

SOME CONCLUSIONS 10b5

Thus in designing a debugger, we have the following goals: 10b5a

1) Define a basic set of core commands that will be useful for a broad set of implementation languages. 10b5b

2) Define implementation language specific commands, and if two or more languages have similar constructs then use the same commands for each of these languages. 10b5c

3) Choose a design for the debugger that allows growth in both additional core commands and the support of new languages. This design goal dictates that the internal structure of the debugger be as follows: 10b5d

There will be a debugger kernel that implements the language-independent features of the debugger and makes calls on language-dependent packages (or modules). 10b5d1

There will be a number of language-dependent packages to implement the language-dependent features. The interface to these packages must be well defined and invarient over time. These packages must be dynamically loadable so

that in any one debugging session, a programmer can move about between implementation languages. 10b5d2

THE NSW DEBUGGER CONVENTIONS 10b6

Within the NSW structure the debugger for any procedural language will consist of the following main components and standard interfaces: 10b6a

DEBUGGER FRONTEND AND GRAMMAR 10b6b

The function of the debugger frontend is to interact with the user to collect and parse commands and display results for the user. The debugger frontend will have its user interaction written in CML and the resulting grammar will drive the standard NSW Frontend Command Language Interpreter. The interface from the debugger frontend to the debugger backend will obey NSW Distributed Programming System Procedure Call Protocol standards. 10b6b1

DEBUGGER BACKEND 10b6c

The debugger backend consists of the following modules. 10b6c1

Debugger Backend Executive 10b6c2

The debugger backend executive handles communication with the frontend and interprets commands it is receiving in the form of procedure calls and makes the appropriate translation to the target machine/operating system and language-dependent modules described below. A standard interface will be defined between the debugger backend executive and these modules so that it can be used with different target machines and languages. 10b6c2a

Target Machine and Operating System Module 10b6c3

The target machine module must translate service requests from the debugger executive into low-level machine and operating system calls to obtain and modify memory cells or operating system state information, generally in bit stream form. Interpretation of these bit streams is done with assistance from the Target Language and Operating System module described below. 10b6c3a

Target Language and Operating System Module 10b6c4

The user would prefer to use symbols at the source language level in communicating with the debugger and in receiving communication from the debugger. 10b6c4a

To do so requires a module that contains language-specific and operating-system-specific information and can perform appropriate table lookup and bit stream translations. One of the tables required is the symbol table for the target program being debugged. A standard interface with the debugger executive is required also. 10b6c4b

Implementation Strategies 10b6c5

The modules associated with the debugger backend may exist on the target system or be distributed with only minimal pieces residing on the target system. The choice will depend on technical and administrative factors. For example, if the target system's operating system supports multiple intercommunicating processes, then implementation of the appropriate subset of the Distributed Programming System could provide a correct communications and target program control environment. If the target operating system cannot be modified or does not support the appropriate features, the bulk of the debugger can be implemented on a system that does have the appropriate features, and a small module can be loaded with the target program to handle local control and communication. 10b6c5a

PERSONNEL: 11

10 JUNE 1975

Proposal For Research
SRI No. ISU 75-94

CONTINUATION OF NATIONAL SOFTWARE WORKS DEVELOPMENTS

Part Two - Contractual Provisions

Prepared for:

Information Processing Techniques Office
Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia,

Attention: William E. Carlson

Part Two--Contractual Provisions 12

I. ESTIMATED TIME AND CHARGES . 12a

It is proposed that the work outlined herein be performed during a period of 9 months commencing 18 July 1975. 12a1

Pursuant to the provisions of ASPR 16-206.2, attached are a cost estimate and support schedules in lieu of the DD Form 633-4. Also enclosed is a signed form complete except as to the "Detailed Discussion of Cost Elements." 12a2

II. REPORTS 12b

The work proposed herein will result in three types of documentation. 12b1

1) Online user documentation for the systems and features proposed. 12b1a

2) Hardcopy user documentation, such as cue cards, to be agreed on with ARPA. 12b1b

3) Commented source code program listings. 12b1c

III. GOVERNMENT-FURNISHED EQUIPMENT 12c

The work proposed herein will require Government-Furnished Equipment (GFE). The equipment required is PDP-10 TENEX computer time on the ARPA-owned, but USC-ISI-operated, computer facility between July 1, 1975, and April 17, 1976. ARC users should be able to get at least 50% of the CPU cycles available to users for at least 16 hours a day 7 days a week. The configuration will be 512K words of memory, at least 3 IBM 3330 type disk drives, and 50,000 pages of file storage for ARC users. Until the 512K words of memory are operable, ARC will have access to 70% of available user CPU cycles. 12c1

ARC will also need access to the ARPANET from SRI IMPs for 2 PDP-11 computers. 12c2

ARC will need terminal access to the ARPANET from TIPs. ARC will be provided source code listings of the ELF operating system for the PDP-11. 12c3

IV. CONTRACT FORM 12d

Because of the nature of the work proposed, it is requested that any contract resulting from this proposal be awarded as a Supplemental Agreement to Contract No. F30602-75-C-0156. 12d1

V. ACCEPTANCE PERIOD 12e

This proposal will remain in effect until 15 July 1975. If consideration of the proposal requires a longer period, the Institute will be glad to consider a request for an extension of time. 12e1

Proposal For Research No. ISU 75-94
Continuation Of National Software Works Developments

COST ESTIMATE FOR PROPOSAL
FOR 9 MONTHS, STARTING 18 JULY 1975
1) NSW Frontend Development
2) NSW Protocol Development and Management
3) NLS as an NSW Tool

Personnel Costs		
Supervision	16,132	
Professional	160,951	
Clerical	1,973	
Total Direct Labor	$ 179,056	
Payroll Burden @ 29%	51,926	
Total Labor and Burden		230,982
Overhead @ 110%		254,080
Total Personnel Costs		$ 485,062
Direct Costs (See Schedules that follow)		
Travel	$ 20,352	
Computer Facility & Terminals	139,538	
Materials and supplies	540	
Communication	1,500	
Documentation and Reports	7,257	
Total Direct Costs		$ 169,187
Total Estimated Cost		$ 654,249
Fixed Fee		52,340
Total Estimated Cost Plus Fixed Fee		$ 706,589

SCHEDULE A
DIRECT LABOR

Direct labor charges are based on the actual salaries for the staff members contemplated for the project work plus a factor of 5.4% of base salary for merit increases during the contract period of performance. The precise factor applied is dependent on the estimated period of performance. Frequency of salary reviews and level of merit increases are in accordance with the Institute's Salary and Wage Payment Policy as published in Topic No. 505 of the SRI Administration Manual and as approved by the Defense Contract Administration Services Region.

SCHEDULE B
OVERHEAD AND PAYROLL BURDEN

These rates have been found acceptable by the Department of Defense for billing and bidding purposes for the calendar year of 1975. We request that these rates not be specifically included in the contract, but rather that the contract provide for reimbursement at billing rates acceptable to the Contracting Officer, subject to retroactive adjustment to fixed rates negotiated on the basis of historical cost data. Included in payroll burden are such costs as vacation, holiday and sick leave pay, social security taxes, and contributions to employee benefit plans.

SCHEDULE C
MATERIALS AND SERVICES

Travel

15 trips, Montgomery, @ $320	=	$ 4,800
2 trips, Washington, D.C., @ $346	=	692
6 trips, Boston, @ $382	=	2,292
2 trips, Los Angeles, at $42	=	84
3 trips, Santa Barbara, at $55	=	165
4 trips, Washington, D.C., Montgomery, at $70	=	280
30 days subsistence in Washington, D.C. @ $42.50	=	1,275
18 days subsistence in Boston @ $38	=	684
208 days subsistence in Other cities @ $30	=	6,240
Auto Rental 256 days @ $15	=	3,840

Air fares are based on prices established in the current Official Airline Guide.
Domestic subsistence rates and travel by private auto are established standards based on cost data submitted to and approved by DCAA.

Communication

This is an estimate of the toll charges for telephone calls during the period or performance.

Materials and Supplies

This is an estimate of the cost of materials and supplies based on previous experience with similar projects.

SCHEDULE D
DOCUMENTATION COSTS

Report costs are estimated on the basis of the number of pages of text and illustrations and the number of copies of reports to be produced, in accordance with the following rates per page:

Editing	$2.55
Composition	2.50
Coordination	.74
Proofreading	.77
Illustration	21.96
Press and Binding	.022 per impression

The following is a breakdown of the estimated cost of report production:

printing, 757 pages at $ 6.56 per page = (including editing, composition, report coordination, proofreading)	$ 4,966
Illustrations, 25 @ $21.96	= 549
Press and bindery at, 65,550 printed pages @ $ .022 per printed page =	1,442
Additional Copying Estimated	300
Total Estimated Documentation Costs	$ 7,257

SCHEDULE E
COMPUTER SUPPORT COSTS

1) PDP-10 TENEX Computer Time

a) July 18, 1975 to April 17, 1976 GFE(see above)
b) July 18, 1975, to April 17, 1976
25 units 6 months *
25 * $3,000/unit = $ 75,000

2) PDP-11 Systems

a) Equipment (lease)

1) PDP-11 Development Machine
$1,930/mo x 9 = 17,370
2) Disk estimate 600/mo x 6 = 3,600
b) Maintenance (8 hrs/d, 5 d/wk)

1) DEC PDP-11,
$675/mo x 9 = 6,075

2) PDP-11 disk 200/mo x 6 = 1,200
Total 28,245

3) Terminals

a) NLS workstations (7)

1) Display (7)
$106/mo x 7 x 9 = 6,678

b) Graphics Terminals
1 Tektronix 4014 and hardcopy unit 880x9 = 7,920
1 Tektronix 4012 300x9 = 2,700
c) TNLS Terminals

1) TI 735's (incl maintenance) (9)
$118.70 x 9x9 = 9,615
2) Maintenance, owned TI's (8)
$20/mo x 8 x 9 = 1,440
3) Acoustic couplers (8)
$16/mo x 8 x 9 = 1,152

d) Modems

1) Dial-up (7)
$36/mo x 4 x 9 = 1,296

e) Leased lines

1) DIA, data
$366/mo x 9 = 3,294
2) DIA, voice
$22/mo x 9 = 198

Total 34,293
4) Miscellaneous (estimated) = 2,000

Subtotal (Items 2-4) $ 64,538

Total (Items 1-4) $139,538

* PDP-10 COMPUTER SUPPORT SUBCONTRACT COSTS:

Charges based on estimates from Bolt Beranek and Newman (BBN) Computer time for this project will come from two sources: BBN and ISI (GFE).

Proposal For Research No. ISU 75-94
Continuation Of National Software Works Developments

(J25088) 23-JUN-75 13:36;;;; Title: Author(s): Richard W. Watson/RWW; Sub-Collections: SRI-ARC; Clerk: DVN; Origin: < WEINBERG, PROP.NLS;7, >, 23-JUN-75 13:00 POOH ;;;; ####;

there once was an augmented fellow
whose lunch she regarded as mellow
but this virtual feast
was ripped off by some beast
and with 25079 she did bellow 1

(J25089) 14-JAN-75 09:18;;;; Title: Author(s): Robert Louis Belleville/RLB2; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLB2;

JEW 14-JAN-75 09:21 25090

Bug in (user-progs,address.rel,)

The ADDRESS user program loads with undefined globals, elicits the message "fst entry nonexistent" from the next edit operation following its use, and in some cases (like just now) clobbers the file in such a way that the PC must be deleted. ADDRESS sould be removed from <USER-PROGS> until fixed; its dangerous. 1

JEW 14-JAN-75 09:21 25090

Bug in (user-progs,address.rel,)

(J25090) 14-JAN-75 09:21;;;; Title: Author(s): James E. (Jim) White/JEW; Distribution: /FDBK([ACTION]) ; Sub-Collections: SRI-ARC FDBK; Clerk: JEW;

DVN 14-JAN-75 10:03 25091

San Juan, Comken, and Business Cards Go to COM; Delays at ISI; DDSI weekend working Hours

On Monday morning 1/13 Larry Day's paper with the file name "San Juan" revised after a previous COM run, Ken Victor's paper on CML revised after two previous COM runs, and Anne Weinbergs business cards, in a file named cards, reached DDSI on tape 003 1

They experienced several delays. The files moved to ISI on Thursday night. It turned out that DDSI had not been returning our tapes after they ran them, so no ARC tapes were available at ISI. 2

The operator would not lend me a tape (journal,25082,). I telephoned Sheri Dolbs at DDSI who had never been told to return the tapes. She said she would have tapes sent over and call me in a couple of hours. She never called. About 3:00 I called. Sheri had gone and no one there knew the whereabouts of the tapes. 3

Early Saturday I linked to ISI. The tapes were there but the tape drive was tied up until 4:00. DDSI has never stated to us a policy about working on weekends, but in the past I have always found some one there from 9-5 Saturday and Sunday and found them amenable when I've asked for work over the weekend. I linked back to ISI at 4:15 waited about 40 minutes while they finished running BSYS, and put the file on tape about 5. 4

As expected, DDSI did not answer at roughly 5:01 when I called. I called DDSI several times during the day Sunday and they did not answer. Monday morning I phoned and they said they would pick up tape 003 and run it tonight. 5

DVN 14-JAN-75 10:03 25091

San Juan, Comken, and Business Cards Go to COM; Delays at ISI; DDSI weekend working Hours

(J25091) 14-JAN-75 10:03;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION]) DPCS([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS;also paper copies to Sheri Dolbs at DDSI and Jerry Pipes at ISI); Clerk: DVN; Origin: < HAMILTON, MORETAPEBUS,NLS;3, >, 14-JAN-75 09:56 DVN ;;;;####;

when the moon comes over the mountain

those of you who have not picked up your W-2 forms can do so now in my office, sandy. 1

SLJ 14-JAN-75 11:17 25092

when the moon comes over the mountain

(J25092) 14-JAN-75 11:17;;;; Title: Author(s): Sandy L. Johnson/SLJ; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: SLJ;

JAKE 14-JAN-75 11:23 25093

Another kleptomaniacal lament

Yesterday lunch...today the large green sorter used by NIC. Has anyone seen or used it lately? And with hope that burns eternal...how about the electric stapler? 1

JAKE 14-JAN-75 11:23 25093

Another kleptomaniacal lament

(J25093) 14-JAN-75 11:23;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /SRI-ARC([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

phone contact with IRS

(DATE) 30 Dec 74 1

(BY) Lieberman 2

(ATTENDEES) 3

Leo Ingleby of IRS 3a

Dick Simpko of IRS 3b

Robert Lieberman (RLL) of SRI-ARC 3c

(MEDIUM) PHONE 4

(WHERE) Place of contact 5

Washington, DC area 5a

(ACTION-ITEMS) 6

Call Bob Griffith or Al Kibat 6a

(DISTRIBUTION) JCN DCE RLL 7

(REFERENCES) 8

(REMARKS) 9

While in Washington DC area I called a friend who works for the IRS. I asked what division within the IRS would be interested in exploratory information systems. 9a

He mentioned that there are three major divisions under the IRS commissioners. They are the Audit, Facility Management, and Technical Divisions. Best bet is the Facility Management division. 9b

The Division director is Leo Ingleby (964-3111). His assistant is Dick Simpko. 9c

I called them. Ingleby listened briefly and said they might be intested but best to talk to Dick Simpko. 9d

Simpko said the best guys to talk to would be Bob Griffith or Al Kibat at (202) 964-3126. They are not in this week but will be back next week. Both are in the Information Systems Branch. 9e

Both Ingleby and Simpko said thay had a distributed community and such systems as NLS might be interesting to them. 9f

phone contact with IRS

(DOCUMENTS) Hard copy given and received 10

(GIVEN) None 10a

(RECEIVED) None 10b

RLL 14-JAN-75 15:07 25094

phone contact with IRS

(J25094) 14-JAN-75 15:07;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /JCN([INFO-ONLY]) DCE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

RLL 14-JAN-75 15:18 25095

Contact report : Fran Greehan re NSF proposal

(DATE) 24 Dec 74 1

(BY) Lieberman 2

(ATTENDEES) 3

Fran Greehan of SRI Marketing 3a

Robert Lieberman (RLL) of SRI-ARC 3b

(MEDIUM) FACE-TO-FACE 4

(WHERE) Place of contact 5

SRI-WASH offices in Rossyln, VA 5a

(ACTION-ITEMS) None 6

(DISTRIBUTION) JCN DCE RLL 7

(REFERENCES) 8

(REMARKS) 9

While at SRI-WASH I talkd to Fran Greehan on the recent NSF proposal that SRI-ARC submitted. 9a

He stated that it would not be appropriate to call NSF. 9b

It general takes 60 to 90 days before they review proposals. 9c

A review panal of noted people in the field read the solicited proposals and return comments to NSF. This is a fairly honest process. 9d

If any additional supportive material should be given, it should be sent very quickly after original submittal or not at all. 9e

(DOCUMENTS) Hard copy given and received 10

(GIVEN) None 10a

(RECEIVED) My copy of the NSF proposal. 10b

RLL 14-JAN-75 15:18 25095

Contact report : Fran Greehan re NSF proposal

(J25095) 14-JAN-75 15:18;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /JCN([INFO-ONLY]) DCE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

File Transfer B6700 - SRI-ARC

This is document describes the process of transferring a test file from the Burroughs B6700 via seven track tape to the SRI-ARC TENEX machine, and then into an NLS file. The procedures and conventions needed to facilitate the general transfer of files are also outlined. 1

Test Case: 2

A B6700 file (A "CANDE" File) was written to 7-track tape using the utility program "DUMPALL". This file had the following characteristics: 2a

7-track 2a1

odd parity 2a2

unlabeled 2a3

BCL Character code (Burroughs version of BCD) 2a4

Unblocked 2a5

80 Characters per record 2a6

Each record of the file consisted of text in the first 72 columns of the record and a sequence number in columns 73-80. There were 132 records in file. 2b

This tape was read into the SRI-ARC machine using the Utility program MTACPY. The resulting file was translated into a TENEX file using the utility program TAPCNV. AT this point one would like to be able to do a "COPY SEQUENTIAL" Command in NLS to make an NLS file, but alas life is not that simple. The only statement delimiters the Copy Sequential command now recognizes are single carraige returns or double carriage returns. This file had neither. 2c

The TECO editor was used to insert a carraige return after each 80 characters, and to remove the sequence numbers from each source record. The actual TECO Command used was: 2d

oj 132<72c8di<CR><CA>><ESC> 2d1

which says something like: 2e

go to the beginning of the file and then perform the following sequence of instructions 132 times; skip 72 characters , delete 8 characters, insert a carraige return. 2e1

Next the NLS Command "Copy Sequential " was used to make an NLS

file, using one carraige return to delimit statements. The resulting file has one NLS statement for each record of the original file. These NLS statements contained a variable number of trailing blanks which makes editing difficult (especially on a line processor or TI type terminal). I therefore wrote a content analysis program, (maynard,deltrsp,ca,) which removes trailing blanks from each statement. The resulting NLS file was reasonable, and usable. 2f

General Case - Recommendations: 3

The kinds of information to be transferred from the B6700 into NLS should be studied carefully. The format and structure of the resulting NLS files should be defined. Once this is done reasonable tape formats can be defined for the tape files. An estimate of the predicted volume of the information transfer is also necessary to decide on the level of effort which is justified on both the source and destination environments. 3a

The first requirement of the format conventions is that there a standard way of delimiting those text strings which should map into a single NLS statement. One possible way to do this is to have each record on the tape map into an NLS statement. Since the B6700 will probably write fixed length records this probably implies a record length of 2000 characters, the maximum statement length in NLS. These records could be padded with blanks if the statement was less than 2000 characters. Although this approach is certainly wasteful of tape it has the advantage that the mapping from records to NLS statements is one-to-one and therefore unambiguous. Another possibility is to define a character (or sequence of characters) which delimits what is to become an NLS statement. This approach has the danger that if the mapping from records to statements is many to one and the records are padded with characters which should not really be part of the text in may be difficult for NLS to remove the unwanted padding. 3b

The second requirement of the format conventions is that there be a standard textual representation of the hierarchical structure of the file. There are many possible ways to accomplish this, for example the the number of leading spaces of each statement can define it's hierarchical level, perhaps relative to the preceding statement. Another possibility is to have "statement numbers" in the file. 3c

Once these format conventions are established special purpose "copy sequential" commands can be written in NLS to bring the files into NLS in the desired format. The creation of these special purpose commands is not a major task. I estimate

approximately 3 man days to code and check out each version. One version is required for each set of tape format conventions. 3d

DSM 14-JAN-75 12:23 25096

File Transfer B6700 - SRI-ARC

(J25096) 14-JAN-75 12:23;;;; Title: Author(s): David S. Maynard/DSM; Distribution: /PWO([ACTION]) DVN([INFO-ONLY]) MAP2([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DSM;

```
FILE program %(110,sav,) (program,rel,) %                                1

   DECLARE nul = 0, bs = 010B, ht = 011B, lf = 012B, vt = 013B, ff =
   014B, cr = 015, del = 177B; % codes for format effectors and
   control characters %                                                  1a

   DECLARE eofflg = 1B9; % end of file flag bit set by GTSTS %           1b

   (program) PROCEDURE;                                                  1c

      LOCAL STRING filnam[40];                                           1c1

      LOCAL stid, da;                                                    1c2

      REF da;                                                            1c3

      &da _ lda ();                                                      1c4

      stid _ orgstid;                                                    1c5

      stid.stfile _ da.dacsp.stfile;                                     1c6

      exmfil (s"testfile.txt", stid);                                    1c7

      RETURN;                                                            1c8

      END.                                                               1c9

   (exmfil) PROCEDURE (filnam, stid);                                    1d

      %                                                                  1d1

      This procedure examines the file FILNAM and reports its
      findings down from statement STID.                                 1d2

      The nature of the examination is to count the number of
      instances of each character, and to determine the maximum
      values for the number of characters on a line and the number of
      lines on a page.                                                   1d3

      %                                                                  1d4

      LOCAL char, ochar, charct, linect, jfn, chartb[128], i, total;     1d5

      LOCAL maxcct, maxlct, quo, rem, ovrstk, ovrchr, ovrlin, crtest;    1d6

      LOCAL STRING outstr[500], errstr[200], outsr1[100],
      outsr2[200];                                                       1d7
```

```
% get the JFN for the file requested %                                  1d8
jfn _ sgtjfn (getgtjflg (read, FALSE, 0), filnam, serrstr);            1d9
% open the file %                                                      1d10
sysopen (jfn, read, chrtyp, serrstr);                                  1d11
FOR i _ 0 UP UNTIL > 127 DO chartb[i] _ 0;                             1d12
maxcct _maxlct _0;                                                     1d13
charct _ linect _ 0;                                                   1d14
ovrchr _ ovrlin _ FALSE;                                               1d15
crtest _ FALSE;                                                        1d16
char _ nul;                                                            1d17
LOOP                                                                   1d18
   BEGIN                                                               1d18a
   ochar _ char;                                                       1d18b
   !bin (jfn);                                                         1d18c
   BUMP  chartb [char _ r2];                                           1d18d
   IF crtest THEN                                                      1d18e
      IF char # LF THEN                                                1d18e1
         BEGIN                                                         1d18e1a
            ovrlin _ TRUE;                                             1d18e1a1
         END;                                                          1d18e1b
   crtest _ FALSE;                                                     1d18f
   CASE char OF                                                        1d18g
      IN [SP, del]; BUMP charct;                                       1d18g1
      =nul;                                                            1d18g2
         BEGIN                                                         1d18g2a
```

```
        !gtsts (jfn);                                   1d18g2a1
        IF r2 .A eofflg # 0 THEN                        1d18g2a2
          BEGIN                                         1d18g2a2a
            chartb [nul] _ chartb [nul] - 1;            1d18g2a2a1
            EXIT LOOP;                                  1d18g2a2a2
          END;                                          1d18g2a2b
      END;                                              1d18g2b
    =bs:                                                1d18g3
      BEGIN                                             1d18g3a
        charct _ charct - 1;                            1d18g3a1
        ovrchr _ TRUE;                                  1d18g3a2
      END;                                              1d18g3b
    =LF: BUMP linect;                                   1d18g4
    =CR:                                                1d18g5
      BEGIN                                             1d18g5a
        IF charct > maxcct THEN maxcct _ charct;        1d18g5a1
        charct _ 0;                                     1d18g5a2
        crtest _ TRUE;                                  1d18g5a3
      END;                                              1d18g5b
    =ff:                                                1d18g6
      BEGIN                                             1d18g6a
        IF linect > maxlct THEN maxlct _ linect;        1d18g6a1
        linect _ 0;                                     1d18g6a2
        IF charct > maxcct THEN maxcct _ charct;        1d18g6a3
        charct _ 0;                                     1d18g6a4
```

```
            END;                                                    1d18g6b
         =ht:                                                       1d18g7
            BEGIN                                                   1d18g7a
               DIV charct / 8, quo, rem;                            1d18g7a1
               charct _ charct + 8 - rem;                           1d18g7a2
            END;                                                    1d18g7b
         =vt:                                                       1d18g8
            BEGIN                                                   1d18g8a
               DIV linect / 8, quo, rem;                            1d18g8a1
               linect _ linect + 8 - rem;                           1d18g8a2
            END;                                                    1d18g8b
         ENDCASE;                                                   1d18g9
      END;                                                          1d18h
   total _ 0;                                                       1d19
   FOR i _ 0 UP UNTIL > 127 DO total _ total +  chartb[i];          1d20
   IF linect > maxlct THEN maxlct _ linect;                         1d21
   IF charct > maxcct THEN maxcct _ charct;                         1d22
   CASE ovrlin OF                                                   1d23
      =FALSE:                                                       1d23a
         CASE ovrchr OF                                             1d23a1
            =FALSE: ovrstk _ 'N;                                    1d23a1a
            =TRUE:  ovrstk _ 'C;                                    1d23a1b
            ENDCASE;                                                1d23a1c
      =TRUE:                                                        1d23b
         CASE ovrchr OF                                             1d23b1
```

```
            =FALSE: ovrstk _ 'L;                                    1d23b1a
            =TRUE:  ovrstk _ 'B;                                    1d23b1b
            ENDCASE;                                                1d23b1c
        ENDCASE;                                                    1d23c
    *outsr1* _ "ACTIVE FORMAT EFFECTORS =  ";                       1d24
    IF chartb[CR] > 0 THEN *outsr1* _ *outsr1*, "<CR> ";            1d25
    IF chartb[LF] > 0 THEN *outsr1* _ *outsr1*, "<LF> ";            1d26
    IF chartb[ff] > 0 THEN *outsr1* _ *outsr1*, "<FF> ";            1d27
    IF chartb[ht] > 0 THEN *outsr1* _ *outsr1*, "<HT> ";            1d28
    IF chartb[vt] > 0 THEN *outsr1* _ *outsr1*, "<VT> ";            1d29
    IF chartb[bs] > 0 THEN *outsr1* _ *outsr1*, "<BS> ";            1d30
    IF chartb[nul] > 0 THEN *outsr2* _ "Number of Nulls in file is
    ", STRING (chartb[nul]), EOL ELSE *outsr2* _ NULL;              1d31
    *outstr* _ "Total characters in file is ", STRING (total), EOL,
    *outsr1*, EOL, "PAGE LENGTH =  ", STRING (maxlct), EOL, "PAGE
    WIDTH =  ", STRING (maxcct), EOL, "Number of Pages =  ", STRING
    (chartb[ff] + (IF ochar = ff THEN 0 ELSE 1)), EOL, *outsr2*,
    "OVERSTRIKING =  ", ovrstk;                                     1d32
    stid _ cis (stid, soutstr, s"d");                               1d33
    *outsr2* _                                                      1d34
        "NUL ", STRING (chartb[0]), EOL,                            1d34a
        "SOH ", STRING (chartb[1]), EOL,                            1d34b
        "STX ", STRING (chartb[2]), EOL,                            1d34c
        "ETX ", STRING (chartb[3]), EOL,                            1d34d
        "EOT ", STRING (chartb[4]), EOL,                            1d34e
        "ENQ ", STRING (chartb[5]), EOL,                            1d34f
        "ACK ", STRING (chartb[6]), EOL,                            1d34g
```

```
    "BEL ", STRING (chartb[7]);                                  1d34h
stid _ cis (stid, $outsr2, $"d");                                1d35
*outsr2* _                                                       1d36
    "BS  ", STRING (chartb[8]), EOL,                             1d36a
    "HT  ", STRING (chartb[9]), EOL,                             1d36b
    "LF  ", STRING (chartb[10]), EOL,                            1d36c
    "VT  ", STRING (chartb[11]), EOL,                            1d36d
    "FF  ", STRING (chartb[12]), EOL,                            1d36e
    "CR  ", STRING (chartb[13]), EOL,                            1d36f
    "SO  ", STRING (chartb[14]), EOL,                            1d36g
    "SI  ", STRING (chartb[15]);                                 1d36h
stid _ cis (stid, $outsr2, $"");                                 1d37
*outsr2* _                                                       1d38
    "DLE ", STRING (chartb[16]), EOL,                            1d38a
    "DC1 ", STRING (chartb[17]), EOL,                            1d38b
    "DC2 ", STRING (chartb[18]), EOL,                            1d38c
    "DC3 ", STRING (chartb[19]), EOL,                            1d38d
    "DC4 ", STRING (chartb[20]), EOL,                            1d38e
    "NAK ", STRING (chartb[21]), EOL,                            1d38f
    "SYN ", STRING (chartb[22]), EOL,                            1d38g
    "ETB ", STRING (chartb[23]);                                 1d38h
stid _ cis (stid, $outsr2, $"");                                 1d39
*outsr2* _                                                       1d40
    "CAN ", STRING (chartb[24]), EOL,                            1d40a
    "EM  ", STRING (chartb[25]), EOL,                            1d40b
```

```
        "SUB ", STRING (chartb[26]), EOL,                          1d40c
        "ESC ", STRING (chartb[27]), EOL,                          1d40d
        "FS  ", STRING (chartb[28]), EOL,                          1d40e
        "GS  ", STRING (chartb[29]), EOL,                          1d40f
        "RS  ", STRING (chartb[30]), EOL,                          1d40g
        "US  ", STRING (chartb[31]), EOL,                          1d40h
        "DEL ", STRING (chartb[127]);                              1d40i
    stid _ cis (stid, soutsr2, $"");                               1d41
    dpset (dspallf, endfil, endfil, endfil);                       1d42
    recred ();                                                     1d43
    sysclose (jfn, $errstr);                                       1d44
    reljfn (jfn);                                                  1d45
    RETURN;                                                        1d46
    END,                                                           1d47
FINISH                                                             1e
```

File Analysis Program Source

(J25097) 14-JAN-75 13:25;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, PROGRAM,NLS;6, >, 20-DEC-74 18:20 JBP ;;;;####;

JoAnn's Report

REPORT ON ANN MENDES JOHNSON

1

This report concerns the grade F given to Ann Mendes Johnson in Painting and Printmaking course 206 (Basic Painting), section 3, Spring semester 1974. 2

There was a question as to whether one of the paintings submitted by Ann Johnson for her final grade was actually her own work. 3

After my final class critique, a student informed me that they had seen what appeared to be the same painting also submitted for Jewett Campbell's final class critique. 4

Upon confering with Jewett Campbell, it seemed as though the same painting was submitted in both classes by different students. Lee Gottschalk was the student who submitted the painting in Jewett Campbell's class. 5

The first time Jewett Campbell and I saw the painting in question was when it was submitted for the final critiques of our respective classes. Neither Jewett Campbell or myself had seen either student work on the painting in class or had seen a specific plan of such a painting. (I had stressed that students work in class, only Ann Johnson continually made excuses -- medical, transportation problems regarding her work, etc. -- as to why she was not able to be in class. During the course of the semester, she was sent an Attendance Warning.) The painting was done in an impersonal technique using spray paints (red, yellow, blue, black, white) and masking tape. The painting was unlike the previous work of either student, though in a Basic Painting class a variance of styles and techniques is not particularly unusual. Lee Gottschalk also submitted for his final critique, a smaller painting done in a similar style as the painting in question. 6

Jewett Campbell and I phoned our respective students to further inquire about the matter. Lee Gottschalk swore that he had made the painting and that his parents could attest to the fact that he had. Ann Johnson said that the painting she submitted in the final critique was at her home, not in the painting racks at VCU as I had indicated in our conversation, and that it must be a coincidence that two similar paintings existed and had spontaneously arisen. She agreed to bring her painting to school on the following day for her previously scheduled appointment with me. 7

Both Jewett Campbell and I met with Ann Johnson. We had the painting claimed to be done by Lee Gottschalk. Ann Johnson brought in a similar painting which she claimed was the very same painting that she had submitted to the final class critique, only that she had done more work on it since the critique. I found it difficult to accept Ann Johnson's account of the matter for the following reasons: 8

1. There was no reason for Ann Johnson to rework the painting. 8a

   a. No negative criticism on the painting was expressed by myself or the class in the final critique. 8a1

   b. It was not necessary for her to show me the painting again for completion of the course. 8a2

   c. I question when she had time to rework the painting during the weekend before finals, as she mentioned that my phone call regarding the matter had upset her studies. 8a3

   d. She was not prone to rework previous paintings, even those with specific problems that needed correcting. 8a4

2. Ann Johnson seemed confused as to her alterations and method of working. 8b

   a. Her claims as to what colors were originally applied and in what order were negated by the tell-tale bleeding through on the back of the painting. 8b1

b. The format of the painting had been changed. The painting was divided into six horizontal sections. I am certain that in the painting submitted in the class critique, the white section was the third section from the top. In the altered painting, the white section was the second section from the top. She seemed confused as to whether the second position was the original position or if she had changed its position when she reworked the painting. 8b2

3. The painting that Ann Johnson brought in had every indication that it was hastily created. 8c

a. The construction of the stretcher was poor and allowed the frame to warp. 8c1

b. The stretched material seemed to be a sheet rather than canvas. 8c2

c. The fabric was poorly stretched on the frame, as seen by the sagging areas in the front of the painting and the tears in the back. 8c3

d. There was not adequate gesso applied before beginning the work, as witnessed by the bleeding on the back of the painting. 8c4

e. The surface quality was inferior and not as rich as in the original painting. 8c5

The student asserted that it would have been impossible for her to find the materials and time to reconstruct a painting in a day's time. 8c6

Jewett Campbell and I thought it best to give Ann Johnson the choice of bringing the matter before a committee review or of accepting an F in the course. We felt that we didn't want to seriously damage the student with possible explusion for this incident by bringing it before the administration. Richard Kevorkian, Chairman of the Painting and Printmaking department, was informed of the incident and how we felt it should be handled. He recommended that a report of the incident be kept in the Painting and Printmaking files, in case the matter came up again in the future. 9

When confronted with the choice, Ann Johnson expressed concern that the incident would be recorded in her files. We told her that I would write a report that would be kept in the Painting and Printmaking files, and that the incident would not be reported to Art Education, her major field. 10

Ann Johnson decided to accept an F for the course on the grounds that it would be easier on her emotional well-being if she took the F and had to repeat the course at a later date, than to have a committee review pending over a period of time and thus having an unsettling effect on her studies and her life. Ann Johnson stated that her acceptance of the grade F was not an admission of guilt. 11

It is my opinion that the painting by Lee Gottschalk was in fact the painting that Ann Johnson submitted as her own work in the final class critique for PAP 206. 12

Included with this report are four slides which were taken to further document the incident. The existence of these slides is unknown to the students involved. 13

JoAnn's Report

(J25098) 14-JAN-75 13:26;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, JK'SREPORT.NLS;3, >, 26-DEC-74 13:43 JBP ;;;;####;

letter mockup

11 December 1974
Stanford Research Institute
Augmentation Research Center
Menlo Park, California 94025

Dave Farber
Department of Information and Computer Science
University of California
Irvine, California 92664

As you will recall i visited the Irvine campus in May of this year in connection with the discussion of a possible faculty position. Our verbal understanding as i recall was that while Irvine could not afford to pay for a cross country trip (i was at that time living in Virginia) it would pay for the additional expenses incurred in extending an otherwise paid for trip to San Francisco to the Los Angeles area. Some time ago i sent you the receipts for these expenses and have not received any response, the amount in question is $75.65.

Sincerely,

Jonathan Postel

letter mockup

(J25099) 14-JAN-75 13:29;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, LETTER,NLS;3, >, 16-DEC-74 15:24 JBP ;;;;####;

garage plan

Concrete Floor	1
Exisiting floor has cracks, would cover with new concrete.	1a
Extend Wiring	2
Existing wall sockets are mid-wall height, would place at near floor level.	2a
Attach Sink to existing sewer	3
Put windows in garage door and wall	4
To alllow more natural light in the garage.	4a
Insulate walls and ceiling	5
To conserve heat.	5a
Cover walls with sheet rock	6
Cover ceiling with ?	7

garage plan

(J25100) 14-JAN-75 13:30;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, GARAGE.NLS;3, >, 20-DEC-74 08:59 JBP ;;;;####;

Annotated message from carlson

Bill:
Here is commented version of your message on batch jobs, some of these questions were answered by your monday phone call with dick and me, i hope to be able to prepare a parallel discussion to yours for comparison -- the main differences are in the break down of functions to particular processes (wm, fe, grammar, tool, etc.), and in which processes touch which kinds of files.
--jon.

1

Date: 12 JAN 1975 1136-PDT 2

From: CARLSON at OFFICE-1 3

Subject: batch tools 4

< CARLSON, BATCH-TOOLS.NLS;2, >, 12-JAN-75 11:26 WEC ;;;; 5

I have a simplified model of batch tools which I use to make decision. 6

** How does this model compare with the model presented in the documents "RJE-MODEL, and BJP by Postel and the notes by Warshall and Millstein ? ** 6a

** What decisions ? ** 6b

** It would be very helpful to have your comments keyed to the previously distributed documents. ** 6c

Plase evaluate the model and, by 16 Jan 75, send a message indicating 7

agreement or identify pitfalls in the model by describing scenarios 8

where it fails, and propose SIMPLE revisions which resolve the pitfalls 9

** Should this suspend progress on the implementation of NSW ? ** 9a

A batch job cannot communicate with the user during execution. 10

** Is this a definition or an attribute of batch jobs shared by other types of jobs ? ** 10a

** Millstein defined the terms BATCH, DETACHED, AND INTERACTIVE in a useful way, lets use his definitions. ** 10b

Background jobs on Multics or other time-sharing systems qualify as 11

batch jobs. 12

** Does "background" include TENEX Detatched Jobs ? ** 12a

The following classes of batch jobs are of interest: 13

Predefined NSW Tools: allow a user talking to the Works Manager to 13a

say the logical equivalent of "execute TESTDATA using CRITERIA as 13b

input and producing MONTHLY as output." CRITERIA and MONTHLY are NSW fills. Optionally, the user might specify a host, ie "execute TESTDATA at UCLA91". 13c

** "Predefined" is a new term to me perhaps a further explaination would be helpful. ** 13c1

** Which 360 should we be getting up to speed on NSW/PCP -- RAND or UCLA ? ** 13c2

The WM will know whether the TBH requires all files to be 13c3

resident before a batch job is submitted, or if it supports 13c4

delayed staging off files. If files must be prestaged, the WM 13c5

will move or create the files and remember the local names. 13c6

** The idea of prestaging vs delayed staging of files is what distingishes BATCH and DETACHED tools in Millstein's document, lets use one set of definitions. ** 13c6a

The WM will know the local name of the tool. It will send a 13c7

message to the TBH of the form "run Local-Tool-Name on 13c8

Local-File-1, Local-File-2, NSW-File-3 producing Local-File-4 and 13c9

NSW-File-5 using TEXT-ARG-1,TEXT-ARG-2." 13c10

** This assumes that it is easy to distinguish local (to what) filenames from NSW filenames -- i for one don't buy that assumption. ** 13c10a

** By now everyone should think in terms of Procedure Call Protocol. The procedure call your "message" maps into is defined in the Batch Job Package (BJP) and is named CRTJOB. 13c10b

Annotated message from carlson

CRTJOB (infiles, outfiles => jobid) 13c10b1

The files in the lists infiles and outfiles are filenames that can be handled by file packages, the batch job package calls on a file pagkage either in the same TBH or another TBH to get the files for input or store the result files. ** 13c10c

If the TBH does not 13c11

support delayed staging, then of course there will be no NSW 13c12

files in the list. Note that since this message is in an NSW 13c13

format, we should easily be able to mark local file names, NSW 13c14

file names, and textual arguments. 13c15

** By "in NSW format" do you mean it is a PCP Call ? ** 13c15a

One implementation (not only one) would have the local tool name 13c16

be a text file or catalogued procedure. The Foreman component in 13c17

the TBH would ask the WORKS MANAGER for a correct local name 13c18

corresponding to each NSW Filename (if there is delayed staging 13c19

of files). The local filenames and the textual arguments would be 13c20

substituted into the control file, which would be given to the 13c21

standard scheduler to be executed at its convenience. The only 13c22

uses I have thought of for textual arguments thus far are run 13c23

time parameters like core size, time limit, priority, etc. 13c24

** What is the "Foreman component" ? ** 13c24a

** The textual arguments you suggest are already handled in every case we know of by parameters in the control file required by the batch processing facility, why should this aspect of host specific job control be replicated in the general purpose batch job package ? ** 13c24b

Annotated message from carlson

The TBH must provide the WM with a job ID. The WM must be able to 13c25

get job status information for a given JOBID. 13c26

** See the CRTJOB and STSJOB procedures specified in the Batch Job Package. ** 13c26a

The TBH must signal the WM whenever a job terinates. 13c27

** An interesting point. To do ths the Works Manager must provide a procedure that a batch job package may call when a job terminates. ** 13c27a

RESPPONSIBILITIES 13c28

COMPASS- define language for invoking tools(the WM command 13c28a

language), provide tool for defining other tools to the WM 13c28b

(CML is part of it, but I don't think all of it), Provide a 13c28c

document telling how to define tools. It must identify 13c28d

ooptions with regard to numbers and attribbutes of input & 13c28e

output files, checking of textual arguments, optional files, 13c28f

warranties, etc. 13c28g

** Anthing to say here ?? ** 13c28g1

TBH Installer- provide a mechanism for accepting WM messages 13c28h

and invoking tools, Create ident/jobid/account card with info 13c28i

sent by WM, provide for status probing, signal WM when tools 13c28j

complete, provide a reasonable way to send output reports onto 13c28k

the ARPANET, Provide a document telling how to install 13c28l

additional tools on that machine. 13c28m

** Shouldn't "a mechanism for accepting WM messages" be a "mechanism for accepting and making PCP Calls". ** 13c28m1

Annotated message from carlson

General Issue: How does the WM know how much space to allocate 13c28n

for output files? COMPASS to take responsibility for 13c28o

formulating and documenting some reasonable answer. 13c28p

** HoW does anybody know ? ** 13c28p1

Sequences of NSW Batch Tools: One can envision jobs consisting of 13d

several "standard" NSW batch tools to be run in succession on the 13e

same TBH. On many hosts, the scheduling algorithm will make it 13f

advantages to have the sequence lumped into a multi-activity job. 13g

Yet the WM should know when each activity completes, and have some 13h

options with regrard to file disposition and conditional tool 13i

invokation. Passing files between activities may also necessitate 13j

control stream changes. 13k

** Why should the works manager notice the jobstep completion for multistep one host jobs ? It may be very difficult to get access to this information in any case. ** 13k1

RResponsibilities: UCLA should take the lead in resolving these 13k2

issues, with inputs from COMPASS and all TBH installers. 13k3

"Perfect" Batch Control Streams: contain only local file names. We 13l

want to discourage these in the NSW, but must provide the capabiltiy 13m

so users don't have to leave the NSW just to type in a few simple 13n

control cards and run a batch job on their own machine. All the TBH 13o

must do is append the ident/jobid/account into to the control stream 13p

and retrieve status and output. 13q

** This dosent seem right to me ??? ** 13q1

Annotated message from carlson

Responsibilities: 13q2

COMPASS: WM must accept a command like "run file at place"h, 13q2a

move the file, signal TBH to invoke it 13q2b

** BY "file" are you now refering to a control file ? ** 13q2b1

TBH Installer: responsible for start-up, status and output 13q2c

reporting. 13q2d

Batch Control Streams Containing NSW Filenames: the user builds a 13r

job control stream ready to run, except he wants to refer to files 13s

by NSW names. In general case, would also want to be able to defer 13t

file movement(not this year). Solution to delayed staging of files 13u

should use same TBH features as for predefined NSW Tools. 13v

** The user wants a nsw-wide control file that is like the existing host specific control files but allows each job step to be executed on a different host. The user can construct such a file with any text editor or perhaps a special control file construction tool. When the user wants to have this control file "executed" a tool is called upon to translate (by calling on the works manager) the nsw filenames to file package file names and to call the appropriate batch job packages for each job step. ** 13v1

Responsibility 13v2

SRI: build an interactive tool which works on typewriter 13v2a

terminals as well as displays and replaces NSW filenames with 13v2b

LOCAL names. Eventually, will instead simply identify some of 13v2c

the names as NSW names and will also be able to handle 13v2d

priority etc. After the substitutions are complete, the tool 13v2e

will invoke the WM to initiate the job 13v2f

** This aside on typewriter terminals and display

terminals is out of place and shows a lack of conviction that the front end will provide means to use a range of terminal classes to use thae same tools. ** 13v2f1

COMPASS and TBH Insjallers are responsible for providing the 13v2g

same capabilities as for "perfect" batch control streams and 13v2h

(eventually) as for NSW defined tools. 13v2i

14

------- 15

Here is a scenario of use of a batch tool which is an elaboration of the discussion contained in the RJE-MODEL document. 16

NOTE 16a

The idea of "remote" job entry -- indeed "remote" anything -- in the National Software Works seems to me to be contradictory to the philosophy of NSW. 16a1

INTRODUCTION 16b

The remote job entry model describes how a primarily batch computing task is prepared and submitted, and how the results of the computation are collected and returned. 16b1

MODEL 16c

First we discuss the entities involved in the process of composing a batch job, having it run, and examining the results. 16c1

The principal entity is a batch processing facility. This is expected to be an existing hardware & software unit that will be only minimally changed to interface to the NSW. 16c1a

Examples of batch job proocessing facilities are the B4700 and the IBM 360. 16c1a1

The NSW talks to the batch processing facility via a procedure package called the Batch Job Package (BJP). 16c1b

The batch job package in a sense referees the flow of information between its PCP callers and the batch processing facility. For example the batch job package colects all the input files that are resident on other

hosts before turning the job over to the batch processing facility, and the batch job package may distribute the result files to other hosts when the job is completed by the batch processing facility. 16c1b1

The Batch Job Package interacts with File Packages (FP) to effect the movement of files to and from the Batch Processing Facility. 16c1c

The call on the batch job package to get a job submitted to a batch processing facility is: 16c1c1

CRTJOB (infiles, outfiles -> jobid) 16c1c1a

The files referenced in infiles and outfiles are named so that the batch job package can get them from and put them into the directories owned by NSW at various hosts and manipulated by file packages. Thus these files are named by "file-package-filenames". 16c1c2

The user sees only NSW-filenames so there must be a language/grammar that controls the users interaction which results in the generation of a create job call on a batch job package. This processing for the user must include the mediation of the NSW-filenames the user supplies into the file-package-filenames included in the create job call. 16c1c3

The files themselves are created and examined using the text editors (e.g. NLS) available in the NSW. 16c1d

Some files that are included in a create job call may be standard library files and from the users point of view part of the system. The user may not even be aware of their existence since their names could be supplied by the grammar internally. 16c1d1

The input files are probably in most cases job control files in a particular batch processing facilities specific job control language. There might be grammars/tools to aid the user in constructing such control files for specific batch processing facilities and applications programs. 16c1d2

The user interacts with the front end. The front end contains a command language interpreter that is driven by a grammar. The particular grammar in use for this user at any time depends on which tool the user is accessing. 16c1e

Annotated message from carlson

A scenario for a user creating, submitting, retrieving, and examining a batch job follows: 16c2

The user interacts with an editing tool to create a source program and to concatenate it with a standard file of job control information particular to the Batch Processing Facility to which it will be submitted. The concatenation is accomplished using regular editing commands (not batch specific commands). 16c2a

The user then interacts with the Works Manager and the Batch Job Package mediated by a grammar to submit the file he has created. The grammar and the Batch Job Package will require enough information from the user that the Batch Job Package can retrieve the input files from File Packages, and store the output files. The Batch Job Package will return an identifier for this job which can be used to request status information at a later time. 16c2b

Some of the information needed to run a batch job could be in a standard file that the user always appends his file to, OR this type of information could be in a separate file that is included by the grammar in the create job call automatically, and the grammar could call on a function to edit a standard file to contain user and run specific parameters such as user-name, priority, run-time-limit. 16c2b1

When the job has been processed the user may use an editing tool to examine the output file. Note that the output files have been stored as specified in File Packages and are thus accessible to tools as permitted by the Works manager. 16c2c

A discussion of multi-host batch jobs. 16c3

Suppose a user wanted to run a series of batch jobs steps where each step was to be carried out on a different host. It is not difficult to envision a NSW-batch-control-language in which one could say things like: 16c3a

"If the previous job step was successful then use its output file WALDO appended to control file DOITTOIT as card input to the batch processing facility ABC and call the printer output file GEORGE". 16c3a1

This requires a tool to "execute" files of this NSW-batch-control-langauge to be written. 16c3b

All this should make clear that a batch tool such as a batch

processing facility is not a special case, that batch and interactive tools are accessed by the user and the internal NSW procedure packages in a consistent manner. 16c4

17

Annotated message from carlson

(J25101) 14-JAN-75 13:37;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, CARLSON,NLS;2, >, 13-JAN-75 13:35 JBP ;;;;####;

RLL 14-JAN-75 15:35 25102

Contact report: the real NAVCOSSACT phone call

My error: this is the rea NAVCOSSACT report with McKenzie, not (25071,) which is the contact report for NAVSEC (Bono).

RLL 14-JAN-75 15:35 25102

Contact report: the real NAVCOSSACT phone call

(DATE) 24 Dec 74 1

(BY) Lieberman 2

(ATTENDEES) 3

Doug McKenzie of NAVCOSSACT 3a

(MEDIUM) PHONE 4

(WHERE) Place of contact 5

Washington, DC area 5a

(ACTION-ITEMS) None 6

(DISTRIBUTION) JCN DCE RLL 7

(REFERENCES) 8

(REMARKS) 9

I spoke to Doug McKenzie of NAVCOSSACT (433-3522) while in the Washington, DC area. He clearly said that NLS at his agency is very much in a dormant state. Perhaps in the late spring some interest could be revived. 9a

I felt he was not very enthusiastic about its prospects in any case. No further talks or demos would be appropriate. 9b

My opinion is that the powers that be must have killed the interest that Doug, at least once, had. 9c

(DOCUMENTS) Hard copy given and received 10

(GIVEN) None 10a

(RECEIVED) None 10b

RLL 14-JAN-75 15:35 25102

Contact report: the real NAVCOSSACT phone call

(J25102) 14-JAN-75 15:35;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /JCN([INFO-ONLY]) DCE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

RLL 14-JAN-75 14:14 25103

BUG: No automatic refreshing of screen after using Reserve Journal numbers command.

It seems that after getting reserve numbers from the sendmail subsystem and having them inserted into statement that appears on the screen that the screen is not refreshed automatically (even tho viewspec u is on). A simple F viewspec does reveal them. 1

RLL 14-JAN-75 14:14 25103

BUG: No automatic refreshing of screen after using Reserve Journal numbers command.

(J25103) 14-JAN-75 14:14;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

who has the 725

who took the 725 that was in the "nic" room last week that susan brought back from the east coast? reward. 1

SLJ 14-JAN-75 14:14 25104

who has the 725

(J25104) 14-JAN-75 14:14;;;; Title: Author(s): Sandy L. Johnson/SLJ; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: SLJ;