TRAINING SUMMARY

IBM/FSD (2 persondays)

IBM Group was trying out NLS as training course

1. Persons (users or not) contacted [uppercase if they have a directory]

   Gray KINNIE, Dick KOPP (Architect), Marv KESSLER

2. COURSE:

   Monday, March 15 - DNLS Course - covered in its entirety with Gray

   Tuesday, March 16 - We discussed topics from the third course that seemed relevant to Gray's work including programs, process branches, a brief look at the Output Processor, and useroptions.

3. ASSISTANCE (other than formal course):

   Tuesday afternoon we tried to share screens with Marv and Karolyn but had difficulty with 8.5. Looks like it was a timing problem due to the fact that Karolyn was enabled.

4. APPLICATION

   This group at IBM is evaluating NLS for its use as an aid to structured programming. Gray will be responsible for judging the quality of the code written primarily by Marv Kessler.

5. ISSUES

   Gray had worked through the Primer and the first 2 sample sessions that accompany the second course because of trouble scheduling training for him. I had talked with him over the phone one day to answer questions he had after working through the items mentioned. His comments during the first day of DNLS training included:

   NLS may be overly rich for programming applications

   He learned something like 5 times faster with a trainer than on his own

   He expected a Jump (to) Statement command in DNLS - Jeanne has had a similar experience when teaching DNLS.

   He was grasping things as they were covered but had forgotten what was covered early in the day by afternoon.

   By noon the second day Gray was saturated and ready to quit learning - one and a half days were enough. I had wanted to spend some time on TNLS and on printing files at RADC but decided that could either be done later if necessary or he could learn from other IBM'ers.

   Gray likes to have "crib" sheets and had a one pager made up with TNLS stuff he had trouble remembering. He decided to do the same with DNLS and I think he will take the online version of the DNLS course and edit it to leave just those things he wants notes of.

He talked about the fact that he and Marv had decided to write programs a little differently than we do at ARC. They will put the END statement at the same level as the PROCEDURE statement instead of a level lower to be consistent with rules of structured programming.

Marv noted that there had been a noticeable degradation of service at Office-1 during the past 2 weeks. He was also interested in starting an index to the L10 Guide because he has trouble finding things in it.

AT&T (3 persondays)

1. Persons (users or not) contacted [uppercase if they have a directory]

John HAYES, Ed PILLARD, Steve Herman, Tony SALINGER, Bob Draper

2. COURSE:

Wednesday, March 17 - John and Steve and I worked through Sections 1 - 8 of the third course.

Thursday, March 18 - John and Steve and I covered Section 9 (Programs) from the third course, process branches and the content analyzer (intro only). During the afternoon (John had a class) Steve and I discussed Section 10 (Useroptions) in the third course as well as spaces for tabs and protection.

Ed Pillard was in and out but mostly out both of those days - I answered some questions he had that were covered in the third course material he missed.

Friday, March 19 - I spent the day covering those parts of the third course that Tony didn't already know about. He was very bright and in some sections I gave an introduction and explained how the documentation was structured and left him to read on his own time (userprograms for example). Some of what we covered was material from the second course that was duplicated in the third course which was good since he's never formally had the second course. In addition to what's in the third course we covered how to write process branches, spaces for tabs and an introduction to the capabilities in the content analyzer.

3. ASSISTANCE (other than formal course):

On Thursday morning John had a temporary secretary in the office to do some typing for him since they have no one to do bulk typing. I spent about a half hour with her during the morning to show her just enough to type stuff in that day for John.

Thursday afternoon when John was in class and after I'd finished working with Steve, I spent about a half hour giving a demo to Bob Draper. Bob is from a different group within AT&T and kept wandering in while I was trying to work with John and Steve. I

put him in touch with Jim Bair because of his interest in human factors but he seems to have broad interests. He isn't part of any of the applications active now and I'm not sure if he is a good person to make contact with from this other group within AT&T - John seemed to think he wasn't particularly important now.

In the process of talking about process branches we wrote branches to move messages in from Tenex etc. both at Somerset and White Plains.

4. APPLICATION

Both Tony and Steve are involved in the Bluebook project. Tony is the project leader and is interested in receiving online status reports from all group members. He has a larger interest in facilitating communication of all types within the group and is planning to use the system largely for that end. Steve has a programming background and will initially be doing status reports as well as possibly one of the chapters for the Bluebook report (his part would be about 18 pages). He may also be a candidate for L10 training.

John Hayes is architect and has other interests besides Bluebook. While I was there he was working on a large amount of information that was a textual description of the items in several PERT listings. A big concern of his is to get his boss to use NLS. His boss had never been available for training when a User Services person was there and can never devote a whole day to something like training. I think I'll suggest to John that he set up some one hour sessions with his boss and myself or another trainer to give him some training in small doses in just the areas he's interested in.

Another area John is interested in is getting some people in their Word Processing Center trained to use NLS so they could tape things they want input or give hardcopy to the WP people and get typing support from them since they have no typing support elsewhere. Two women who run (?) the Center attended one of Priscilla's sessions. The session was specifically geared to helping people get started on the status reports so was not particularly applicable to the WP women. They probably need a demo at some future point to see the capabilities of the system.

5. ISSUES

They seem to be doing well for a new client partly because of their enthusiasm and their willingness to allocate time to learning NLS and working at finding applications.

John favored me with his conclusion that ARC was great except for marketing and documentation. They would like to see a user's manual and more consistency between documents.

DMA (1 personday)

1. Persons (users or not) contacted [uppercase if they have a

directory]

Chuck HALL, Tom CHRYST, Don MCENTEE

2. COURSE:

no formal course

3. ASSISTANCE (other than formal course):

The day began with a short briefing by Chuck about DMA. There are 6 groups within this directorate that together produce maps for the Air Force. Chuck is in Plans, Requirements, and Technology. That group is responsible for handling the transition of software from an R & D state into an operational environment. Tom is in the Cartography Department where a lot of systems analysis is done. Don is in the Research Department where people have backgrounds in math and programming. Other departments are the Missile Department where they have an in-house text editor - so therefore feel they have no application for NLS, and the Aeronautical Information Department where the FLIPs are generated. When Chuck was at ARC the FLIPs looked like they might be a type of documentation that would be a good application for NLS. The last department is the Printing and Distribution Department that prints and distributes maps etc. - no applications apparent here.

We then talked about what each person was interested in using NLS for so I'd know what to spend the time talking about. We got started in Useroptions and changed printoptions, feedback indenting and the default viewspecs to make things look better to them. We discussed the archival schedule and how that effects the way a person works.

Eventually it became apparent that they did have a few things that would possibly be applications of NLS. They brought samples and we typed in a memo without utilizing structure or any directives outside the origin in order to make it as simple as possible (it has a relatively simple format anyway). Directives in the origin included those to print a 10 1/2" page, leave top margin etc. I made notes which were a subset of the material from the first course so Tom could take away a list of only those commands necessary to do a memo or possibly documentation. I then got a copy of the TDY memo format to bring here in order to set up a template for them to use.

They said a xerographically reproduced copy would be acceptable for the final copy of such memos and TDY reports so I suggested they xerographically reproduce the TI paper - we tried it and they thought it would be acceptable.

I worked with Chuck a little to redo a process branch written for him by Bobbie Carrier when he was visiting RADC. It didn't sort messages brought in from Tenex. Chuck and I also looked through the Lineprocessor Users Guide with attention to the section on setting up the workstation and logging in so when they get their line sometime soon he will be able to check out the equipment.

4. APPLICATION

They don't really have one yet. They have done some communicating with people they have contracts with at RADC. I worked with them a little and will try to get Chuck and Tom a procedure for typing TDY reports online. There is a possibility that Tom will use the system for documenting programs and procedures he is responsible for.

They are expecting ARC to do some sleuthing for applications at some point in the future to come up with some really good things.

5. ISSUES

Part of the problem has been that DMA originally used Autovon lines to the RML TIP beginning last fall. When the RML TIP was moved to Gunter they no longer had access so therefore no way to use OFFICE-1. They have recently obtained permission to call Autovon to Gunter allowing them to use OFFICE-1 after 3-4 months of non-use. Within the next 2 weeks they are to have a line to the RADC TIP which will allow them to use the Display (still unpacked) and TI.

The Autovon lines are very noisy and they had a lot problems with CTRL-X's coming across the lines and aborting commands - I've since sent a message suggesting they change their Useroptions to ignore DEL.

BLANK FORM

CLIENT (# personday)

1. Persons (users or not) contacted [uppercase if they have a directory]

2. COURSE:

3. ASSISTANCE (other than formal course):

4. APPLICATION

5. ISSUES

BLANK CONTACT REPORT

CONTACT: journal-title

DATE: date

BY: name-of-contactor

ATTENDEES: Full name of organization, address, phone number as substatements

Name of attendee - Organization acronym

MEDIUM: PHONE/LETTER/COMPUTER/CONFERENCE/FACE-TO-FACE

WHERE: Place-of-contact

ACTION-ITEMS:

Actions taken, to be taken, etc., dated

DISTRIBUTION: ARC-LOG DCE JCN RLL

REFERENCES:

DOCUMENTS: Hard copy given and received

GIVEN: Date and documents given

RECEIVED: Date and documents received

REMARKS:

ENCRYPTION

< JJOURNAL, 27955.NLS;1, >, 27-Apr-76 14:39 XXX ;;;; .HJOURNAL="KEV 27-APR-76 10:11 27955"; Title: .HI="encryption subsystem available for initial release"; Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /SRI-ARC([INFO-ONLY]) LAC([INFO-ONLY]) JEG([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: KEV; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

.PEL; .PN=PN-1; .GCR;this document describes the 2 commands available in the encryption subsystem and a description of the encryption process I am releasing for initial shakedown a new encryption subsystem. The subsystem lives at ISIC as (xprograms, encrypt.subsys,) and (xprograms, encrypt.cml,). The source code is also at ISIC as (xprograms, encrypt.nls,). The subsystem consists of the following two commands (I beleive it to be bug free, but as with any new software, it may take a few days to verify this; this initial release is a subset of the final implementation to be made available within the next 7-9 weeks):

Encrypt file

This command asks the user to (twice) specify a (non-echoed) key. The user specified key consists of from 1 to 100 characters (using NLS PASSWORD selection algorithm). The user specified key is then converted to a 36 bit value which is used as the key for encrypting the file (and its PC if one exists) loaded in the current display window. (The final implementation may convert the user specified key to a 72 bit value, but we haven't come to a final decision on this yet.)

The created encrypted files will exist in the same directories as the files of which they are an encryption. They will have the same filename and version numbers as the source files, but will have the extension names of ENC (for the NLS file) and EPC (for the PC if one existed).

The encrypted files will contain an encryption of the key used to encrypt them, an encrypted page table, an encryption of the user settable word from the FDB (the user settable word in the FDB of the encrypted files will be 0), and if applicable and encryption of the filename for the encrypted PC.

In the final implementation, the source files will be marked for deletion after they are encrypted.

Load encrypted file

This command accepts a filename and a key. The key is checked against the key in the specified encrypted file.

(In the final implementation, a user will be given 5 attempts to specify the correct key for one file, or 10 total incorrect attempts per session, before being logged out with messages sent to the last writer of each file for which an incorrect key was specified.)

When a correct key is specified an NLS file will be created that is a decipherment of the encrypted file. It will have the same filename as the encrypted file and an extension of NLS and a version number of one greater than the version number of the encrypted file. If the encrypted file contained an encrypted user settable word, then a deciphered PC will be generated and loaded.

An attempt to load an encrypted file via the normal NLS mechanisms of Load file or Jump to link will generate the message that the specified file is not an NLS file.

Algorithm

The following is a loose discussion of the algorithm presented for anyone interested.

First some definitions.

A key refers to a 36 bit value.

We will use the term cipher to refer to a bit stream of random bits. The ciphers we use vary in length between 36 bits (one word) and 512*36 bits (one page) long. A cipher is generated by placing a 36 bit key in a feedback shift register (FSR). The FSR is shifted 6 bits at a time. (This shifting of the FSR is not a straightforward logical shift, but includes feeding back some of the bits in the FSR as the FSR is shifted.) Three bits are then selected from the FSR. One of these three bits is then used to select which of the other 2 bits to use to generate one bit of output for the cipher.

(More details on the operation of the FSR and cipher generation are available either from the source code itself, or from Dave Hopper (ARC).)

(GETCSTREAM is the name of the procedure used to generate ciphers. GETCSTREAM generates the requested number of cipher bits based on a passed key, and returns the final contents of the FSR. This resulting value (of the FSR) is sometimes used as a key for future cipher generation.)

We frequently have need for a random number in some specified range. To get the required random number, a 36 bit cipher is generated (based on varous keys). The resulitng cipher modulo the requested range and added to the base number in the range produces the requested random number.

(RANDNUM is the procedure used to generate a random number in the passed range based on a passed key. The procedures INITRANDOM, DONERANDOM, and EXCLRANDOM are used when a group of unique random numbers are needed in a range (e.g. for randomizing the pages of a file). INITRANDOM is called first, being passed the range and initial key. It returns a data structure. EXCLRANDOM is called each time a new unique random number in the requested range is needed. It is passed the address of the data structure generated by INITRANDOM. EXCLRANDOM uses the key in the data structure to generate a randum number (call it N) and then returns the Nth unused number in the requested range. The key in the data structure is then updated to be the resulting value of the FSR for the next use of EXCLRANDOM. DONERANDOM is used to free up the allocated data structure.)

We will use the terms encrypted data, encryption, (or similar terms) to refer to the bit pattern that results from the exclusive or of source data with a cipher.

Noise pages refer to duplicate copies of pages in the source file that are not used in the decryption of the file.

I think we are now ready to discuss how a file is encrypted. (The encryption is performed by procedure ENCFILE.)

First the user specified string is converted to a key by the following:

The key is initialized to 0.

For each character in the user string, the key is rotated left 5 bits and the user specified character is exclusive or'ed with the rotated key.

The user specified key is used to generate keys that are used for

the rest of the encryption process. These keys are words in the cipher stream produced by using the user specified key. Call these keys FK[i].

(In the final implementation, we may generate a 72 bit key based on the user string. This 72 bit key will be used to generate 36 bit FK[i].)

One of these FK is used to determine a random number in the range 0-NPLIMIT. Call this number GP as it represents to the number of noise pages that will be generated. NPLIMIT is currently set to 20.

Two of these FK are used to represent the value of the user specified key that is kept in the resulting encrypted file.

One of these FK is used to generate unique random numbers in the range 0 to TP-1. TP is the number of pages that will exist in the resulting encrypted file. It is the sum of SP (the number of pages in the source file) plus GP plus one (for the key page, i.e. the page containing the user key representation, etc.). These random numbers are used to scramble the order of the pages, i.e. the pages in the resulting encrypted file are generated in the following order:

First, the key page is placed in a random page in the encrypted file.

Then, the noise pages, if any, are placed in random pages in the encrypted file.

Next, the first existing page of the source file is placed in a random page in the encrypted file.

Next, the second existing page of the source file is placed in a random page in the encrypted file.

And so on, until all pages in the source file have been placed in the encrypted file.

One of these FK is used to generate a number of keys that are used for the encrypting of each page. Call these keys PK[i].

Each page to be encrypted uses a different set of PK[i]. The initial set of PK[i] are generated from one of the FK[i]. Successive PK[i] are generated by the resulting value of the FSR after the generation of a set of PK[i].

Each page of the resulting encrypted file is produced as follows:

Key Page:

A unique destination page number (called EPI), i.e. the random page in the encrypted file, is selected.

A set of PK[i] are generated for this page.

A random source file page (based on a PK[i]) is selected and mapped in (into page SFBUFF).

A random index into the source page is selected (based on a PK[i]).

Starting at this random location (and wrapping around if necessary), the user settable word from the FDB of the source file, the 2 appropriate FK[i], a representation of a page table for the source file, and if appropriate, a string containing the name of the encrypted PC, are placed in the source page.

A 512*36 bit cipher (based on a PK[i]) is generated and placed into page CKPAG.

The source page is rotated a random number of words (based on a PK[i]) and exclusive or'ed with the cipher (into page CKPAG).

The resulting page is then mapped out to the destination encrypted file.

Noise Pages:
A unique destination page number (called EPI), i.e. the random page in the encrypted file, is selected.
A set of PK[i] are generated for this page.
A random source file page (based on a PK[i]) is selected and mapped in (into page SFBUFF).
A 512#36 bit cipher (based on a PK[i]) is generated and placed into page CKPAG.
The source page is rotated a random number of words (based on a PK[i]) and exclusive or'ed with the cipher (into page CKPAG).
The resulting page is then mapped out to the destination encrypted file.

Source File Pages:
A unique destination page number (called EPI), i.e. the random page in the encrypted file, is selected.
A set of PK[i] are generated for this page.
The appropriate source file page is mapped in (into page SFBUFF).
A 512#36 bit cipher (based on a PK[i]) is generated and placed into page CKPAG.
The source page is rotated a random number of words (based on a PK[i]) and exclusive or'ed with the cipher (into page CKPAG).
The resulting page is then mapped out to the destination encrypted file.

And finally, the user settable word of the FDB of the encrypted file is set to 0.

The source file in encrypted first, and then the partial copy (if one exists) is encrypted using the final value of the FSR which is returned by ENCFILE.

RADC - Proposal

HGL RWW KEV RLB2 DSM 21-MAY-76 15:45 28125

Initial Suggestions for General Support Projects for RADC in FY 1977

Dick Watson will be at RADC next week and will be prepared to discuss the suggestions mentioned in this memo. If there are any comments or questions before then, please contact either Dick or Harvey Lehtman at ARC.

Initial Suggestions for General Support Projects for RADC in FY 1977

Introduction 1

This document is submitted to the Rome Air Development Center (RADC) by the Augmentation Research Center (ARC) of SRI; it outlines some possible developments to and research on some aspects of NLS and programming technology which might be carried out beginning in fiscal year 1977. It must be stressed that this is not a formal proposal, but is rather a discussion of technical issues as we currently view them. Interaction with the RADC staff will, we hope, lead to a refinement of the discussion and, finally, a formal proposal of work of mutual interest and benefit. As this is basically a thinkpiece submitted to elicit comments, cost and research estimates are only approximate. After further discussion with RADC, we hope to submit a formal proposal to execute those items of greatest interest; this formal proposal will have more detailed work and cost breakdowns. 1a

The suggestions cover 6 major areas of interest: 1b

Widening Availability of NLS: NLS on Other Computer Systems 1b1

Graphics Subsystem Maintenance and Extensions 1b2

Software Engineering Tools Development 1b3

Online Software Debugging Facilities: Research and Development 1b4

Project and Configuration Management Tools 1b5

Data Management Systems and NLS 1b6

We feel especially strongly that putting NLS on a minicomputer configuration will offer large advantages; our reasons are outlined in detail below. Even if a full funding for this task is not possible at this time, we feel the initial design study discussed in the following section would prove valuable. 1c

It should be stressed that ARC is willing to consider other suggestions for research about which RADC feels strongly. 1d

Widening Availability of NLS: NLS on Other Computer Systems 2

Introduction 2a

With small effort, NLS will be converted this summer to the DEC 2040, the first in DEC's new System 20 family of computers. The DEC supported operating system for this line, TOPS 20, is derived from TENEX. We estimate that NLS in this configuration will be about 1.5 times faster than the current Office-1

machine. The cost of a 2040 configuration capable of simultaneously supporting about 25 NLS users will be about $450K. This available machine with manufacturer supported hardware and operating system able to support NLS thus may serve as a baseline against which we may compare other alternative methods of providing NLS service. 2a1

We see three principal motivations for RADC's consideration of other configurations for use with NLS: 2a2

1) To obtain NLS on an in house machine to reduce the cost of its increasing internal NLS usage. 2a2a

2) To provide a computer environment to aid in the technology transfer of NLS technology for Air Force and DoD applications such as documentation, communication, and software development of vital concern to RADC's charter. 2a2b

3) To support ARC's ongoing research and development of NLS and to insure the most cost effective use of evolving technology in that development. 2a2c

There are two paths we may take in moving NLS from the PDP-10 or System 20 environment to other hosts: movement to other large scale timesharing systems such as MULTICS or transfer to a minicomputer environment. In the following, we discuss why we feel it would be advantageous at the present time to begin the development of NLS on as minicomputer based system to achieve the maximum medium to long range cost savings and to provide the most effective user environment. 2a3

Moving NLS to an environment such as MULTICS is difficult to justify solely for RADC internal use because the development cost of such a move is close to the purchase price of a System 20. However, if there are significant technology transfer possibilities, then this move may be fully justified and should be pursued. For example, if the MULTICS environment is going to play an important role in WWMCCS software development and maintenance or in other important Air Force application domains, then RADC might make a significant contribution by assisting in the move of NLS directly into that environment. 2a4

In the NSW environment it may still be possible for developers of Honeywell 6000 application systems to make use of NLS on DEC hardware for the Air Force applications without moving it directly to MULTICS. This is clearly one of the goals of NSW and, if security considerations would allow such a possibility, we should pursue this approach with RADC. The cost of developing tools to effectively work in this mode might be

substantially less than the cost of moving NLS directly to MULTICS. 2a5

In the following sections, we discuss in greater detail, first, the work required to move NLS to MULTICS and other large scale timesharing systems and, second, the work necessary to move NLS to a minicomputer environment and the current importance of such a move. 2a6

NLS on MULTICS or Other Large Timesharing Systems 2b

NLS is cleanly split, in the NLS-9 version, into Frontend and Backend components. The Frontend handles all user interface functions and terminal control. It parses user commands and calls information processing routines in the Backend for execution. Currently the Frontend runs on PDP-10 and PDP-11 computers. It requires full duplex character-at-a-time interaction with the user to provide the full features of the NLS human/machine interface. The Frontend and Backend do not have to run on the same machine and there are economic arguments why it may be preferable to have the Frontend on a minicomputer if the Backend is on a large timesharing system. 2b1

Though the MULTICS operating system supports file system and other features necessary to implement the NLS Backend, its line-at-a-time half duplex terminal interface is not suitable for the NLS Frontend. Use of NLS with MULTICS would therefore be through a Frontend machine such as the PDP 11 with an NLS Backend resident on MULTICS. 2b2

We estimate the development cost of moving the NLS Backend to MULTICS to be about 5-6 person years of effort costing about $400K, including computer support. In addition, there would be the cost of a PDP-11 Frontend Computer. Depending on the exact configuration selected, this Frontend machine would cost in the neighborhood of $70K - $120K. 2b3

It is difficult to estimate the relative performace of NLS running in the MULTICS environment versus running in the PDP 10 environment at this time, although we would expect NLS to run faster on MULTICS because of its larger memory and faster CPU. 2b4

The tasks required to move NLS to MULTICS are the following: 2b4a

1) Produce a PDP-10 to MULTICS cross compiler for the L10 language in which NLS is currently written. 2b4a1

2) Design an Operating System interface for NLS to make

NLS as independent as possible of a specific operating system. 2b4a2

By following this approach, we get the added benefit of being able to easily move NLS to other backend computers in the future with much less cost. 2b4a2a

3) Reorganize NLS to interface to the Operating System Interface. 2b4a3

4) Learn the MULTICS Operating System and implement the MULTICS side of the Operating System Interface. 2b4a4

5) Make the general cleanup of NLS code that would be required as a result of such a reorganization. 2b4a5

6) Set up maintenance procedures for multiple host versions of NLS. 2b4a6

7) Provide appropriate system documentation. 2b4a7

8) The issue of how to provide user program capability if the L10 compiler is on a PDP-10 would have to be addressed. It may be possible to provide a linking to the NLS environment from some other language already existing on MULTICS. Otherwise, a version of the L10 compiler may have to be created to operate on MULTICS. 2b4a8

Movement of NLS to MULTICS would require about 18 months calendar time minimum. We think 24 months is probably more realistic. 2b4b

NLS on a Minicomputer Configuration 2c

We believe that the most cost effective way to provide NLS service to users in the medium to long range is to provide each user with a work station including not only a display, keyboard, pointing device, etc., but also processing power and file storage to support the basic functions of NLS. We feel this is true for the following reasons: 2c1

1) The cost of large scale integrated circuits for logic and memory is decreasing rapidly. It is conservatively estimated that the cost per bit or gate for such memory or logic will decrease in cost by a factor of 100 in the next ten years. 2c1a

2) Communication costs, while likely to decrease, will not decrease at anywhere near the same rate. To take advantage

of the low logic and memory costs will therefore require computing capability local to the user. 2c1b

The reasons for this lag in the decrease of communications cost are due in part to the large capital investment in existing facilities and in the regulatory environment of communications. 2c1b1

Communication costs will be one of the prime factors causing local computing to pull ahead of central remote large shared processors as the most cost effective configuration. Other factors are discussed below. 2c1b2

Note, however, that there will still be tools and services that the user will want to access at remote sites, but his most commonly used tools should be local to his work station. 2c1b3

3) Significant improvements in responsiveness, user interface services, and information portrayal will become possible with display technology developments. These capabilities can be tapped only with the high bandwidth available through local computing. 2c1c

To take advantage of bit mapped video display technology, for example, will in itself require considerable local memory and processor capability. The additional cost to provide processing power for NLS functions is incrementally small. 2c1c1

4) While the decreasing logic and memory costs will also apply to large scale processors, we feel market pressures and economies of manufacturing scale will be most pronounced at the minicomputer end of the computer spectrum. Further use of central large scale timesharing systems will involve the communication delays and costs mentioned above. Moreover, the operating system complexity and overhead is greatly reduced for a one man per CPU system.. 2c1d

5) With his own CPU, new and valuable services can operate in the background when the user is not interacting with the system. 2c1e

6) Large scale processors require special environmental care such as air conditioning, high (and possibly backup) power and a staff of operators. These services are not be required with a minicomputer based system in the user's office. 2c1f

7) A distributed minicomputer based system will be more reliable. 2c1g

Besides the potential to significantly improve the cost effectiveness of NLS service for use in RADC and elsewhere in the Air Force, the existence of standalone (or networks of standalone) NLS systems can be applied in new Air Force field applications not presently easily accommodated with an NLS system based on a large central machine. It also means that NLS can be applied in applications in smaller start up cost increments and that expansion can be more easily handled on a one workstation at a time basis. 2c2

We envision the following model: Each user would have a workstation with a powerful CPU (of the PDP 11/34 class), at least 64K of memory, 5-10 million bytes of bulk store, a high performance bit mapped mixed text and graphics video display and appropriate input devices. This system would be connected to the ARPANET for access to databases and tools not local to his work station and for interperson communication. The connection point might be another minicomputer with a large disk store or the capability to perform other functions for a local group of users such as RADC. Note that the exact details of how to most effectively provide the amount of file storage required and the network connection would need detailed study and design. However, this description gives a flavor of the possibilities. 2c3

The appropriate first steps for which we seek funds are to configure a PDP-11 system in a manner similar to that outlined above, to move the NLS BASE subsystem to it, and to make the necessary changes in the PDP-10 version of NLS to perform file and other function sharing for those capabilities infrequently used or for which a larger central machine, by its nature, is the most appropriate location. The PDP-11 is a type of minicomputer system which we feel will be commonly available at greatly reduced costs in the next five years; we would probably use the UNIX operating system for the PDP-11. 2c4

We estimate that about $50K worth of hardware would be purchased for a prototype workstation. About 4-5 person years (about $275K to $350K) over 18 months worth of development would be required to bring up the first prototype. The rapidly decreasing hardware cost should allow this workstation to be operationally available for $5K to $10K in the medium term period of 3-5 years. The following is a rough work breakdown: 2c5

1) Decide on the correct language environment for such a system. We would need either to extend the current L10 to

PDP-11 compiler to support the full L10 language orto produce an interpreter. The latter approach has much to offer for such a system. 2c5a

2) Choose and learn a new operating system. The UNIX operating system is a likely candidate. 2c5b

3) Measure and analyze the use and structure of the NLS code to see how it should be organized in the minicomputer address space enviroment. 2c5c

4) Design and implement an operating system interface. 2c5d

5) Decide whether a version of NLS for both the minicomputer and large timesharing system environments is feasible or whether two different diverging versions would be required. 2c5e

6) Provide appropriate system documentation. 2c5f

7) Make the code modifications or rewrites required. 2c5g

We feel that starting toward such a minicomputer NLS system is very important. Even if funds to make the full first steps to an operational use of the BASE subsystem are not available, the smaller step of measuring and analyzing the use and structure of the NLS code and performing preliminary design studies is useful and should be pursued. Such a program of analysis could be started with 6-12 person months of support. 2c6

Once the BASE Subsystem is operational, further developments would involve evaluation and planning aimed at providing this type of system to RADC and other NLS users. We must also then decide which other NLS subsystems to move into the minicomputer environment. 2c7

Graphics Subsystem Maintenance and Extensions 3

Graphics System Maintenance 3a

The graphics system associated with NLS is very large and complex. Its performance to date has been, we think, remarkable both in the quickness with which the system was developed and the stability and general usefulness of the package. 3a1

The newness and flux of the system requires both intermittent maintenance and ongoing tuning both internally and in the user interface. 3a2

An expanding body of users discovers minor bugs in any system. The cost of repairing these troubles is not inconsequential. Some errors of the scope of those which have been discovered are to be expected: they are intrinsic to systems of this complexity. What is remarkable is that there are so few! 3a3

More important than these errors and bugs are new features which are motivated by new users. User requirements vary widely as do user idiosyncrasies. Minor system modifications and improvements are an important part of the evolution of a system like Graphics. 3a4

Currently there are no explicit support funds for this type of maintenance and improvement. Without such support the Graphics system maintenance and improvement will continue to be slow and sporadic. We heartily recommend that 4 man-months over a 12 month period be allocated to the job of supporting and improving the Graphics subsystem. 3a5

Data Presentation 3b

The current graphics system forms a basis for the construction of several types of auxiliary "graphics" systems. We believe that the most important is data presentation: the production of graphic illustration and charts from raw data. 3b1

Persons involved in the documentation or criticism of material frequently preprocess numeric and tabular data into tables, illustrations and various charts. This processing is relatively difficult and involves the following steps: 3b2

1) Assembly of relevant data 3b2a

NLS is a powerful tool in the assembly and scanning of online data. However, this task is diverse and time consuming. 3b2a1

2) Selection of the Form of presentation 3b2b

The user selects a standard method such as graphs, barcharts, piecharts, or other methods appropriate to the job and style of the problem. 3b2b1

3) Processing of Data 3b2c

Methods involve statistical analysis, estimation, approximation, and extrapolation. Computation packages exist for many types of analysis. 3b2c1

4) Scaling of data 3b2d

5) Drawing of standard portions of the plots 3b2e

These include axes, scales and grids. 3b2e1

6) Drawing of curves and creation of symbols 3b2f

7) Captioning and labeling of the plots 3b2g

The data presentation package would make the creation of graphics and charts more automatic by: 3b3

1) Assisting in the capture and grouping of data 3b3a

Interface to existing analysis packages in FORTRAN and other languages is possible. 3b3a1

2) Automating the scaling of data 3b3b

3) Providing direct commands for the creation of axes, etc. 3b3c

4) Providing direct commands for the plotting of lines and such symbols as bars on barcharts and sections of pie in piecharts. 3b3d

Application of the data presentation package will enable users to: 3b4

1) Plot more often and at lower cost. 3b4a

2) Plot more uniformly with improvements in the overall quality of illustrated material. 3b4b

Data presentation provides a useful and needed addition the the existing Graphics subsystem. The development, design, implementation, testing and documentation would require 8.0 man-months. 3b5

Software Engineering Tools Development 4

Response to IBM Evaluation and Suggestions and Development of Online Structured Programming Environment 4a

The current IBM-FSD contract with RADC is concerned with the evaluation of the NLS environment's suitability for top down structured programming techniques. ARC has a parallel contract with RADC to assist the IBM group and to provide them with instruction in our techniques to insure a fair evaluation based

on as much knowledge as possible. We expect that among the results of these contracts will be a group of suggestions leading to improvements in the power of NLS to better support top down structured programming. 4a1

Additionally, through our years of experience with NLS and top down structured programming techniques, ARC has accumulated a number of its own ideas for modifying the NLS programming environment. 4a2

Examples include additions to our implementation languages and the creation of an NLS PSL (Programming Support Library) subsystem. 4a2a

For this contract, we would recommend that 12 man months be allocated to two major tasks in this area: 4a3

(1) Integrate ARC's ideas with the results of the above contracts and create an ordered list of additions and modifications to the NLS programming tools to better support top down structured programming. This list would include an estimate of the resources needed to implement each of the suggestions. 4a3a

(2) Implement a number of the most valuable suggestions. 4a3b

Online Software Debugging Facilities: Research and Development 5

Introduction 5a

As part of its ongoing development of a Software Engineering Augmentation System providing interactive online tools for software development fully integrated into the NLS environment, ARC has for several years been concerned with the development of sophisticated debugging techniques. Extensions to the standard Dynamic Debugging Technique (DDT) programs available on our operating systems lead to NDDT, a system with a user interface consistant with NLS standards and fully aware of the ARC programming language and runtime environments. Extensions and developments have recently resulted in a multi-process, multi-machine debugger for use in the NSW. This debugger has been modularized for ease of integration into operating systems other than TENEX and languages other than L10 with the development of new language and operating system modules. We propose research and development work under this contract which would extend the state of the art of debugging techniques: the extension of the NSW debugger to the PL/1 or JOVIAL languages on MULTICS and research into the debugger of the future. 5a1

Debugging PL/1 or JOVIAL on MULTICS in an NSW Environment 5b

ARC has developed a pilot online interactive debugger to operate in an NSW environment. It is organized into a number of modules: a general purpose Debugger Dispatcher (DD) module; a Language Module (LM) that contains all the semantic and syntactic knowledge of the language in which a program is written; and an Operating System Module (OSM) that contains all the knowledge of the operating system and machine upon which a program to be debugged is running. 5b1

We are completing LMs and OSMs to debug NSW programs of the following types: L10 programs running on a TENEX or an ELF Tool Bearing Host (TBH), and BCPL programs running on a TENEX TBH. 5b2

The model will work to provide an interactive debugging tool for other environments in the NSW. As a demonstration of this fact and as an attempt to have other programmers working in this preferable mode, we propose to provide the debugging tool to programmers at RADC who write PL/1 or JOVIAL programs to be run on MULTICS. 5b3

To extend the debugger to be able to debug programs in a language other than L10 running on a MULTICS TBH in an NSW environment requires a LM for that language, a MULTICS OSM, and a package to live on MULTICS to communicate with the MULTICS OSM. We expect that providing these modules would take approximately 9 man months as follows: 5b4

3-5 man weeks to learn about MULTICS PL/1, JOVIAL or other language 5b4a

9-13 man weeks to write a PL/1, JOVIAL or other language LM 5b4b

9-12 man weeks to learn about MULTICS and write the MULTICS communication package 5b4c

7-10 man weeks to write the MULTICS OSM 5b4d

Advanced Online Interactive Debugging Tool Research 5c

What should the debugger of the future look like? Though studies indicate that more time spent in the design phase of a project leads to less time devoted to debugging, we feel enhanced debugging capabilities will be desirable in all types of programming projects: implementation errors will still occur. 5c1

The complexity of programs will increase with new machine

architectures and the advent of truly distribted, multi-machine and multi-process systems. 5c1a

There are several approaches to the debugging problem. Some of these are sketched below: 5c2

1. Non-interactive debugger development. 5c2a

   While non-interactive tools are, we feel, inferior to an integrated online environment, they will still be needed in certain configurations. Tools could be created for use in these situations. For example, more flexible trace facilities, more meanigfully formatted dumps, and post-dump or post-trace analyzers may be valuable. 5c2a1

2. Interactive debugging. 5c2b

   A comparison of an integrated programming environment (with the debugger tied directly to the languages and editors used as in many LISP systems) with more general purpose modular debuggers is one of several issues which may be explored. 5c2b1

3. Graphical debuggers. 5c2c

   Such systems could display and permit manipulation of graphical representations of program contexts and data structures. 5c2c1

4. Source level debuggers. 5c2d

Another important set of questions deals with the cost effectiveness of the varying techniques. 5c3

We propose that 3 man months be allocated to study possible approaches to the design of the advanced debugger leading to a report summarizing existing debugging techniques and the techniques likely to appear in the next few years. 5c4

In addition, we propose that 6 man months be allocated to the implementation of some of the features that look most promising. These features would be implemented, on an experimental research basis, using the debugger framework we have developed at ARC for the NSW. 5c5

For example, one of the features to be implemented might be a source language debugging facility for one specific language. 5c5a

Project and Configuration Management Tools 6

NLS as an Aid to Management and Coordination of Research and Development Projects 6a

NLS has proven itself to be a valuable tool in several areas: as an aid in high quality document production; as the center for the recorded dialog of a collaborating, geographically distributed group; as a data collection and dissemination center for information about the ARPANET; as the base for various database management applications; and as the keystone for a set of advanced tools for software production, development, debugging and documentation. 6a1

NLS has also proven useful in several aspects of internal ARC management: proposal creation, progress report creation and dissemination, document production (for the creation of proposals, final reports, and other project documentation), and the use of various calculating tools for accounting are some examples. Integration with the NLS Graphics subsystem and with various data management programs, either those within NLS or those available externally, could lead to flexible and dynamic report generation for management of both individual projects and contract offices responsible for the coordination of many contractors or subcontractors. A configuration management package could be created for this coordination. PERT techniques could be introduced as well. 6a2

As research and development projects increase in cost and complexity, efficient and timely management techniques become more critical. More and more projects are collaborative in nature: automated tools for the coordination of development work of several subcontractors by contract officers become essential. 6a3

It would be presumptuous for us to suggest detailed designs for such tools without knowing how the end users (managers in project offices, for example) currently view their roles and tasks. Conversely, it would be difficult to obtain sophisticated, educated ideas for such developments from these users if they are not aware of what is currently available in the NLS (and network) environment. 6a4

We therefore propose the following tasks for the current contract period which would total 1 man year of effort: 6a5

Augment a project officer (or group of officers). He would be instructed in the uses of the current version of NLS for his work. Techniques for entry of necessary information

would be "packaged" and presented in an integrated manner. The introduction of NLS into this environment would be handled in a more controlled manner than is usual with the introduction into the office of a typical utility client. We would interact closely with the selected project manager(s) to discover what tools they feel would be most useful in an online environment. We would study how they currently do their work with an eye to possible additions to NLS. 6a5a

Survey currently available (automated and non-automated) project management tools. We would consider their addition to the NLS Workshop environment either directly or by interface and prepare a design for such an introduction in a possible extension of the contract. 6a5b

Implement an interface to a basic package of management tools from currently existant NLS features. 6a5c

Data Management Systems and NLS 7

NLS and Data Management 7a

NLS is a very effective tool for a wide variety of applications: for example, its power in report creation is clear. It also provides a rich environment for the creation of tools for specialized applications. This generality, however, has served to limit its effectiveness as the base for data management systems operating over large, highly structured data bases. While a number of data management applications have been developed in the NLS workshop and have been successfully used, these applications have been constrained to medium sized data bases. The generalized NLS file structure so useful in other applications does not provide optimal data management support such as multiple access methods and multiple orderings on a given set of records. In addition, the generalized file structure cannot match the efficiency obtainable by tailoring the file structure to a given application. 7a1

Examples of recent data management applications in NLS include the following: 7a1a

QUERY II 7a1a1

An interactive system capable of querying a data base of naval ships and ports. Extensive search and updating capabilities included. Developed for a demonstration of NLS data management capabilities for the Naval Electronics Laboratory. 7a1a1a

FMS and DES 7a1a2

A tool for maintaining a data base of information concerning the status of ongoing projects. Retrieval and report generation facilities are included. Developed jointly by SRI-ARC and RADC as a project management subsystem. 7a1a2a

GMARK and CIMROS 7a1a3

Two NLS subsystems developed at SRI for coordinating government and commercial marketing activities. These subsystems provide application tailored data collection, data modification, query, and report generation capabilitites. 7a1a3a

There are many systems specialized to data base management (DBMS's). These systems are designed to effectively manage large structured data bases. In general these systems offer some or all of the following facilities: 7a2

A high level language for defining the data base records, their logical structure and access requirements. 7a2a

A high level language for defining the physical file structures for the data base. 7a2b

A query language for interrogating a data base. 7a2c

A report generation language. 7a2d

An interface to one or more programming langauges which make the data base accessible to applications programs. 7a2e

Some of these DBMS's run on existing ARPANET hosts, and a few of these run on machines which will in the near future operate within the framework of the National Software Works (NSW) project. The user interfaces of most of these systems are syntactically unique and often dificult to master. Report generation and formatting capabilities are often inadequate. These inadequacies are contrasted with NLS's strengths in these areas. 7a3

There are many possible designs for providing users with the combined capabilities of NLS and a DBMS. Some of the design variables are: 7a4

1) The requirements of the particular data management application, or class of applications. 7a4a

2) The characteristics of the particular DBMS. 7a4b

3) The degree of integration of the DBMS host computer system into the ARPANET/NSW environment. 7a4c

We propose that SRI-ARC perform a detailed study of alternative designs for the integration of a data management application, or set of applications, specified by RADC. This study will be a 3 man-month effort and will produce a report detailing the design alternatives, associated costs, and our design recommendations. Implementation, contingent upon RADC's acceptance of the proposed design and costs, could be accomplished through extensions to the contract covering the design study. 7a5

OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFI
OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFI
OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFICE-1 OFFI

MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD
MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD
MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD MAYNARD

(DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DS
(DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DS
(DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DSM)28442 (DS

TUESDAY, SEPTEMBER 28, 1976 10:21:43-PDT TUESDAY, SEPTEMBER 28, 1976 10:21:43-P
TUESDAY, SEPTEMBER 28, 1976 10:21:43-PDT TUESDAY, SEPTEMBER 28, 1976 10:21:43-P
TUESDAY, SEPTEMBER 28, 1976 10:21:43-PDT TUESDAY, SEPTEMBER 28, 1976 10:21:43-P

< KJOURNAL, 28442.NLS;1, >, 29-JUL-76 17:51 XXX ;;;; Title:
Author(s): Harvey G. Lehtman, Kenneth E. (Ken) Victor/HGL KEV;
Distribution: /JLM([ACTION]) DLS([ACTION]) RAL([ACTION])
SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: HGL;
Origin: < LEHTMAN, PROP76.NLS;16, >, 29-JUL-76 11:45 HGL ;;;;####;

Suggestions for General Support Projects for RADC in 1977

This is a revised version of the earlier document (28125,) based on conversations between ARC and RADC.

Suggestions for General Support Projects for RADC in 1977

INTRODUCTION 1

This document is addressed to the Rome Air Development Center (RADC) by the Augmentation Research Center (ARC) of SRI; it outlines some possible extensions to and research on aspects of NLS and programming technology which might be carried out beginning in calendar year 1977. This is not a formal proposal, but is rather a discussion of technical issues as we currently view them. Interaction with the RADC staff will, we hope, lead to a refinement of the discussion and to a formal proposal for work of mutual interest and benefit. As this is basically a thinkpiece submitted to elicit comments, cost and research estimates are only approximate. After further discussion with RADC, we hope to submit a formal proposal to execute those items of greatest interest; this formal proposal will have more detailed work and cost breakdowns. 1a

The suggestions cover 4 major areas of interest: 1b

Software Engineering Tools Development 1b1

Online Software Debugging Facilities: Research and Development 1b2

Graphics Subsystem Maintenance and Extensions 1b3

Widening Availability of NLS: NLS on Other Computer Systems 1b4

It should be stressed that ARC is willing to consider other suggestions for research in which RADC is interested. 1c

SOFTWARE ENGINEERING TOOLS DEVELOPMENT 2

Response to IBM Evaluation and Suggestions and Development of Online Structured Programming Environment 2a

The current IBM-FSD contract with RADC is concerned with the evaluation of the NLS environment's suitability for top down structured programming techniques. ARC has a parallel contract with RADC to assist the IBM group and to provide them with instruction in our techniques to insure a fair evaluation based on as much knowledge as possible. We expect that among the results of these contracts will be a group of suggestions to extend NLS's usefulness in structured program development. 2a1

ARC has developed many tools and techniques, fully integrated into NLS, for aiding groups of programmers developing large, complex, interrelated systems. These tools include language writing systems and the structured languages written in them, interactive debugging tools, code cataloging and library

systems, the use of programming development teams, the development and use of system-wide coding and documenting standards, and tools for supporting communication between collaborating workers on a programming team. Of course, the online environment available to all NLS users with its powerful editor, heirarchically structred files, and user programming facilities is especially valuable to programmers. 2a2

For this contract, we would recommend that 14 man months be allocated to two major tasks in this area: 2a3

1. Review the IBM Structured Programming Series prepared for RADC and the IBM-FSD evaluation of NLS's programming environment. Prepare an ordered list of desirable additions and modifications to the NLS family of tools to better support top down structured programming. This list would include an estimate of the resources needed to implement each of the suggestions. 2a3a

2. Implement a number of the most valuable suggestions jointly selected by RADC and ARC as time permits. 2a3b

ONLINE SOFTWARE DEBUGGING FACILITIES: RESEARCH AND DEVELOPMENT 3

Introduction 3a

ARC has been creating a set of interactive online tools for software development for many years. This Software Engineering Augmentation System is fully integrated into the NLS environment. As part of this effort, ARC has investigated, implemented, and used sophisticated debugging techniques. Extensions to the standard Dynamic Debugging Technique (DDT) programs available on our operating systems lead to NDDT, a system with a user interface consistant with NLS standards and fully aware of the ARC programming language and runtime environments. Extensions and developments have recently resulted in a multi-process, multi-machine debugger for use in the NSW. This debugger has been modularized so that it may be easily integrated into operating systems other than TENEX and languages other than the ARC language L10 by developing new language and operating system modules. 3a1

Research and development work under this contract would extend the state of the art of debugging techniques in two areas: 3a2

Investigation of the use of the ARC debugger in a JOVIAL production environment. This would involve the extension of the debugger to a subset of the JOVIAL language on a TENEX machine as a demonstration of the feasibility of the use of

the ARC debugger model for Air Force production work. We would also prepare a design for the introduction of a JOVIAL debugger into an Air Force software production environment running on a GECOS machine connected to the ARPA network. 3a2a

Research into the nature of the debugging process, examination of debugging tools currently available in both research and production environments, and consideration of the debugger of the future. 3a2b

The tasks proposed in these two areas are discussed in the following sections. 3a3

Demonstration JOVIAL Interactive Debugger and JOVIAL Network Debugger Design 3b

ARC has developed a pilot online interactive debugger to operate in an NSW environment. It is organized into a number of modules: a general purpose Debugger Dispatcher (DD) module; a Language Module (LM) that contains the semantic and syntactic knowledge of the language in which a program is written; and an Operating System Module (OSM) that contains knowledge of the operating system and machine upon which a program to be debugged is running. 3b1

We are completing LMs and OSMs to debug NSW programs of the following types: L10 programs running on a TENEX or an ELF Tool Bearing Host (TBH), and BCPL programs running on a TENEX TBH. The debugger itself, while capable of debugging programs running on a variety of machines, is itself designed to run on a TENEX, either as a standalone TENEX subsystem or within the NSW environment. 3b2

The existing NLS/NSW interactive debugger framework can handle other languages and operating systems through the creation of the appropriate language and operating system modules and communication environment. 3b3

ARC has made use of its debugger in its research and development work. As a demonstration of the system and in order to make the power of our debugger more widely available in a real world code production environment, we propose to provide a demonstration version of the debugging tool for JOVIAL programs to be run on a TENEX machine. As a demonstration, an LM for the JOVIAL subset which currently runs on a PDP10 will be created. 3b4

The JOVIAL compiler running on a TOPS-10 will be brought

into TENEX for this demonstration. The demonstration debugger will operate with JOVIAL programs on TENEX 3b4a

We will also prepare a design study of the possible implementation of the JOVIAL debugger for a JOVIAL compiler operating on GECOS in the ARPANET. 3b5

There are several important issues to be considered in such a design. These issues include the following: 3b5a

1. Creation of an Operating System Module for GECOS and the interface of the system to a non-DEC computer. The system could be distributed with the core debugger functions running on the PDP10 and the object programs running on the GECOS machine, or a version of the system could be developed to run completely under GECOS. 3b5a1

2. Debugging mode for batch programs. The optimal facilities for debugging programs which are developed to run in a batch environment must be provided. It is currently unclear how this may be best accomplished while still maintaining the power of the system for the debugging of interactive programs. While GECOS provides some interactive capabilities, we assume many of the programs under development are to be run in batch mode. 3b5a2

We expect that the preparation of the demonstration JOVIAL debugger and the design for network integration will take a total of approximately 10 man months as follows: 3b6

Demonstration project: 6 man months 3b6a

1-3 man months to learn about JOVIAL 3b6a1

3-5 man months to write a LM to support (a reasonable subset of) JOVIAL 3b6a2

Network JOVIAL debugger design: 4 man months 3b6b

Study of the Debugging Process and Present and Future Debugging Tools 3c

What should the debugger of the future look like? Though studies indicate that more time spent in the design phase of a project leads to less time devoted to debugging, enhanced debugging capabilities will be desirable in all types of programming projects: implementation errors will still occur. 3c1

The complexity of programs will increase with new machine

architectures and the advent of truly distributed, multi-machine and multi-process systems. 3c1a

There are several approaches to the debugging problem. Some of these are sketched below: 3c2

1. Non-interactive debugger development. 3c2a

   While non-interactive tools are, we feel, inferior to an integrated online environment, they will still be needed in certain configurations. Tools could be created for use in these situations. For example, more flexible trace facilities, more meaningfully formatted dumps, and post-dump or post-trace analyzers may be valuable for finding bugs and also for code analysis and optimization. 3c2a1

2. Interactive debugging. 3c2b

   A comparison of an integrated programming environment (with the debugger tied directly to the languages and editors used as in many LISP systems) with more general purpose modular debuggers is one of several issues which may be explored. 3c2b1

3. Graphical debuggers. 3c2c

   Such systems could display and permit manipulation of graphical representations of program contexts and data structures. 3c2c1

4. Source level debuggers. 3c2d

Another important set of questions deals with the cost effectiveness of the varying techniques. 3c3

We propose that 6 man months be allocated for a 2 part study: 3c4

The first part would be an examination of the nature of software debugging and a survey of existing debugging techniques used at a number of different sites and institutions (including several Air Force installations). 3c4a

The second part would be a survey of the research work currently being performed in the area of debugging which will affect future debugging systems. A summary of the debugging techniques likey to be available in the near future on a production basis will be included. 3c4b

In addition, we propose that 6 man months be allocated to the

implementation of some of the features that look most promising. These features would be implemented, on an experimental research basis, using the debugger framework we have developed at ARC for the NSW. 3c5

GRAPHICS SUBSYSTEM MAINTENANCE AND EXTENSIONS 4

Graphics System Maintenance 4a

The graphics system associated with NLS has been useful and reliable in several initial applications. As a complex, new, and changing system, Graphics has required intermittent maintenance and ongoing tuning of its core software functions and user interface. 4a1

An expanding body of users discovers minor bugs in any system. The cost of repairing these troubles is not inconsequential. Some errors of the scope of those which have been discovered are to be expected: they are intrinsic to systems of this complexity. 4a2

More important than these errors and bugs are new features which are motivated by new users. User requirements vary widely as do user idiosyncrasies. Minor system modifications and improvements are an important part of the evolution of a system like Graphics. 4a3

We propose support funding for 4 man-months over a 12 month period be allocated for maintenance and improvement of the Graphics subsystem. 4a4

Data Presentation 4b

The current graphics system forms the basis for constructing several auxiliary systems. The most important of these is data presentation: the production of graphic illustrations and charts from raw data. 4b1

Persons involved in the documentation or criticism of material frequently preprocess numeric and tabular data into tables, illustrations and various charts. This processing is relatively difficult and involves the following steps: 4b2

1) Assembly of relevant data 4b2a

2) Selection of the presentation form 4b2b

The user selects a standard method such as graphs,

barcharts, piecharts, or other methods appropriate to the job and style of the problem. 4b2b1

3) Processing of Data 4b2c

Methods involve statistical analysis, estimation, approximation, and extrapolation. Computation packages exist for many types of analysis. 4b2c1

4) Scaling of data 4b2d

5) Drawing of standard portions of the plots 4b2e

6) Drawing of curves and creation of symbols 4b2f

7) Captioning and labeling of the plots 4b2g

The data presentation package would make the creation of graphics and charts more automatic by: 4b3

1) Assisting in the capture and grouping of data 4b3a

Interface to existing analysis packages in FORTRAN and other languages is possible. 4b3a1

2) Automating the scaling of data 4b3b

3) Providing direct commands for the creation of axes, etc. 4b3c

4) Providing direct commands for the plotting of lines and such symbols as bars on barcharts and sections of pie in piecharts. 4b3d

Application of the data presentation package will enable users to: 4b4

1) Plot more often and at lower cost. 4b4a

2) Plot more uniformly with improvements in the overall quality of illustrated material. 4b4b

Data presentation provides a useful and needed addition to the existing Graphics subsystem. The development, design, implementation, testing and documentation would require 8.0 man-months. 4b5

WIDENING AVAILABILITY OF NLS: NLS ON OTHER COMPUTER SYSTEMS 5

Introduction 5a

With little effort, NLS will be converted this summer to the DEC 2040, the first in DEC's new System 20 family of computers. The DEC supported operating system for this line, TOPS 20, is derived from TENEX. We estimate that NLS in this configuration will be about 1.5 times faster than the current Office-1 machine. The cost of a 2040 configuration capable of simultaneously supporting about 25 NLS users will be about $450K. This available machine, with manufacturer supported hardware and operating system able to support NLS, thus may serve as a baseline against which we may compare other alternative methods of providing NLS service. 5a1

We see three principal motivations for RADC's consideration of other configurations for use with NLS: 5a2

1) To obtain NLS on an in house machine to reduce the cost of its increasing internal NLS usage. 5a2a

2) To provide a computer environment to aid in the technology transfer of NLS technology for Air Force and DoD applications such as documentation, communication, and software development of vital concern to RADC's charter. 5a2b

3) To support ARC's ongoing research and development of NLS and to insure the most cost effective use of evolving technology in that development. 5a2c

Two alternatives for moving NLS from the PDP-10 or System 20 environment to other hosts are: movement to other large scale timesharing systems such as MULTICS or transfer to a minicomputer environment. The following discusses the advantages of developing a version of NLS on a minicomputer based system. This development provides the maximum medium-to-long range cost savings and offers the most effective user environment. 5a3

Moving NLS to an environment such as MULTICS is difficult to justify solely for RADC internal use because the development cost of such a move is close to the purchase price of a System 20. 5a4

In the NSW environment it may still be possible for developers of Honeywell 6000 application systems to make use of NLS on DEC hardware for the Air Force applications without moving it directly to MULTICS. This is clearly one of the goals of NSW and, if security considerations would allow such a possibility, we should pursue this approach with RADC. The cost of developing tools to effectively work in this mode might be substantially less than the cost of

moving NLS directly to MULTICS. (A design study or trial implementation linking the MULTICS and DEC/NLS environments in the NSW could be included in this project if desired by RADC.) 5a4a

In the following sections, we discuss in greater detail, first, the work required to move NLS to MULTICS and other large scale timesharing systems and, second, the work necessary to move NLS to a minicomputer environment and the current importance of laying the groundwork for such a move. 5a5

NLS on MULTICS or Other Large Timesharing Systems 5b

NLS is cleanly split, in the NLS-9 version, into Frontend and Backend components. The Frontend handles all user interface functions and terminal control. It parses user commands and calls information processing routines in the Backend for execution. Currently the Frontend runs on PDP-10 and PDP-11 computers. It requires full duplex character-at-a-time interaction with the user to provide the full features of the NLS human/machine interface. The Frontend and Backend do not have to run on the same machine and there are economic arguments for running the Frontend on a minicomputer if the Backend is on a large timesharing system. 5b1

Though the MULTICS operating system supports file system and other features necessary to implement the NLS Backend, its line-at-a-time half duplex terminal interface is not suitable for the NLS Frontend. Use of NLS with MULTICS would therefore be through a Frontend machine such as the PDP 11 with an NLS Backend resident on MULTICS. 5b2

We estimate the development cost of moving the NLS Backend to MULTICS to be about 5-6 person years of effort costing about $400K, including computer support. In addition, there would be the cost of a PDP-11 Frontend Computer. Depending on the exact configuration selected, this Frontend machine would cost in the neighborhood of $70K - $120K. 5b3

The high cost of moving the NLS Backend to MULTICS in comparison with the cost of implementing an NLS Minicomputer configuration (see below) or purchasing a DEC 2040 based NLS suggest that this is not a cost-effective approach at this time. 5b4

NLS on a Minicomputer Configuration 5c

The most cost effective way to provide NLS service to users in the medium to long range is to provide each user with a work

station including not only a display, keyboard, pointing device, etc., but also processing power and file storage to support the basic functions of NLS. Arguments supporting this contention follow: 5c1

1) The cost of large scale integrated circuits for logic and memory is decreasing rapidly. It is conservatively estimated that the cost per bit or gate for such memory or logic will decrease in cost by a factor of 100 in the next ten years. 5c1a

2) Communication costs, while likely to decrease, will not decrease at anywhere near the same rate. To take advantage of the low logic and memory costs will therefore require computing capability local to the user. 5c1b

The reasons for this lag in the decrease of communications cost are due in part to the large capital investment in existing facilities and in the regulatory environment of communications. 5c1b1

Communication costs will be one of the prime factors causing local computing to pull ahead of large central, remote, shared processors as the most cost effective configuration. Other factors are discussed below. 5c1b2

Note, however, that there will still be tools and services that the user will want to access at remote sites, but his most commonly used tools should be local to his work station. 5c1b3

3) Significant improvements in responsiveness, user interface services, and information portrayal will become possible with display technology developments. These capabilities can be tapped only with the high bandwidth available through local computing. 5c1c

To take advantage of bit mapped video display technology, for example, will in itself require considerable local memory and processor capability. The additional cost to provide processing power for NLS functions is incrementally small. 5c1c1

4) While the decreasing logic and memory costs will also apply to large scale processors, we feel market pressures and economies of manufacturing scale will be most pronounced at the minicomputer end of the computer spectrum. Further use of central large scale timesharing systems will involve the communication delays and costs mentioned above.

Moreover, the operating system complexity and overhead is greatly reduced for a one man per CPU system. 5c1d

5) With his own CPU, new and valuable services can operate in the background when the user is not interacting with the system. 5c1e

6) Large scale processors require special environmental care such as air conditioning, high (and possibily backup) power and a staff of operators. These services are not required with a minicomputer based system in the user's office. 5c1f

Besides the potential for improving the cost effectiveness of NLS service for use in RADC and elsewhere in the Air Force, the existence of standalone (or networks of standalone) NLS systems can be applied in new Air Force field applications not presently easily accommodated with an NLS system based on a large central machine. Also, such a configuration will permit NLS applicatons with smaller initial cost increments. 5c2

The NLS minicomputer configuration could follow this model: Each user would have a workstation with a powerful CPU (of the PDP 11/34 class), at least 64K of memory, 5-10 million bytes of bulk store, a high performance bit mapped mixed text and graphics video display and appropriate input devices. This system would be connected to the ARPANET for access to databases and tools not local to his work station and for interperson communication. The connection point might be another minicomputer with a large disk store or the capability to perform other functions for a local group of users such as RADC. Note that the exact details of how to most effectively provide the amount of file storage required and the network connection would need detailed study and design. However, this description gives a flavor of the possibilities. 5c3

We estimate that, were such a system to be implemented today, about $50K worth of hardware would be required for a workstation. However, the rapidly decreasing cost of hardware should allow this workstation to be operationally available for $5K to $10K in the medium term period of 3-5 years. 5c4

For this contract we propose one man year of effort to study and measure the current NLS code and produce a preliminary design for a prototype NLS minicomputer based system which provides the essential editing capabilities of the current BASE subsystem. The following subtasks would be included: 5c5

1) Measure and analyze the use and structure of the NLS code

to see how it should be organized in the minicomputer address space environment. 5c5a

2) Decide whether a version of NLS for both the minicomputer and large timesharing system environments is feasible or whether two different diverging versions would be required. 5c5b

3) Decide on the correct language environment for such a system. Should current L10 to PDP-11 compiler be extended to support the full L10 language or should we produce an interpreter? 5c5c

4) Choose and learn a new operating system. The UNIX operating system is a likely candidate. 5c5d

We estimate that, given such a study, the first prototype could be implemented in 3-4 additional person years. 5c6

SUMMARY OF PROPOSED WORK 6

The following summarizes the tasks outlined above. The times and costs are rough estimates and would be analyzed in greater detail in our formal proposal. Cost estimates include computer support and SRI overhead charges. The total estimated time for the proposed work is 60 person months for an estimated cost of about $400,000. 6a

SOFTWARE ENGINEERING TOOLS DEVELOPMENT -- 14 person months total 6a1

1. Review the IBM Structured Programming document and the IBM-FSD evaluation of NLS. Prepare a document outlining possible additions to NLS to aid in structured programming. -- 4 person months 6a1a

2. Implement a number of the most valuable suggestions jointly selected by RADC and ARC as time permits. -- 10 person months 6a1b

ONLINE SOFTWARE DEBUGGING FACILITIES: RESEARCH AND DEVELOPMENT -- 22 person months total 6a2

1. NLS Demonstration Debugger for JOVIAL and Design for Network JOVIAL Debugger. -- 10 person months 6a2a

a. Demonstration debugger. -- 6 person months 6a2a1

b. Network debugger design. -- 4 person months 6a2a2

2. Debugging study and survey. -- 12 person months 6a2b

a. Survey and examination of debugging systems. -- 6 person months 6a2b1

b. Feature implementation -- 6 person months 6a2b2

GRAPHICS SUBSYSTEM MAINTENANCE AND EXTENSIONS -- 12 person months total 6a3

1. Graphics system maintenance and improvement. -- 4 person months 6a3a

2. Data Presentation package. -- 8 person months. 6a3b

WIDENING AVAILABILITY OF NLS: NLS ON OTHER COMPUTER SYSTEMS -- 12 person months total 6a4

1. NLS on a minicomputer: design study. -- 12 person months 6a4a

INTRODUCTION

The purpose of the Stand Alone Front End (SAFE) is to provide the L10 programmer (or, with modifications, programmers of other languages) with a powerful, low-overhead user interface that resides in the same fork as the execution (backend) procedures. The rest of this document is concerned with the L10 version of SAFE; i.e. all references to procedure calls, returns, etc. are assumed to obey L10 call/return/etc. conventions.

FUNCTION of SAFE

According to the associated grammar, the execution procedures will be provided with arguments and called. They may do arbitrary processing and must return a result indicating success or failure. Other results may be returned as well.

SAFE also provides procedures callable by the backend to show information or error strings to the user, obtain information from him, etc.

SAFE INTERFACE to BACKEND

Type of Interface

SAFE interfaces to the Backend through procedure calls. These calls are dispatched through one of two mechanisms: SAFE dispatching or Backend dispatching. The choice of mechanism is up to the Backend writer. (see "MAKING a STAND ALONE TOOL")

SAFE Dispatching

The simpler of the two mechanisms is SAFE dispatching. With it, Backend procedures are called directly with the arguments as specified in the grammar and one additional argument. This additional argument is the address of an L10 list that may be used to return results to the Frontend.

Example:

CML function call in grammar:

```
fun( cmlvar1, cmlvar2)
```

Corresponding L10 procedure call made by SAFE to Backend:

```
fun( var1, var2, resultlist REF);
```

where var and var2 are copies of the cmlvar1 and cmlvar2 and resultlist is empty.

Backend Dispatching

A backend may need to perform some function each time one of its procedures is called from SAFE, eg. do argument conversion or address resolution. SAFE provides for this by allowing the

backend to do its own procedure calling.

The backend must supply a dispatcher for this purpose. The dispatcher will be called by SAFE whenever the grammar specifies that a backend procedure call should be made. It's arguments will be the address of a string containing the name of the procedure to be called, the address of a list of arguments for the procedure, and the address of a list in which to place results. The dispatcher is expected to call the procedure before returning to SAFE.

Example:

CML function call in grammar:

```
fun( var1, var2)
```

Corresponding L10 procedure call made by SAFE to Backend:

```
dispatcher( $"fun", $arglist, $resultlist);
```

where arglist contains the values of var1 and var2 and resultlist is empty.

Success/Failure Indication and Returning Results

SAFE expects the Backend procedure it calls to return as its primary result TRUE if the call was successful or FALSE if it was not. This is the case regardless of whether SAFE calls the specific Backend procedure or it calls a backend dispatcher, that is, regardless of whether SAFE dispatching or Backend dispatching is being used.

If the Backend call was successful, then results (up to 8) put into resultlist by the Backend will be assigned to the corresponding CML builtin variables RESULT1 through RESULT8. If resultlist is empty, then RESULT1 through RESULT8 are set to NULL.

If the Backend call was not successful, SAFE expects to find in resultlist an error number and user-readable string as the first and second elements respectively. These will be shown to the user by SAFE and then the command will be aborted.

Calling procedures in SAFE

SAFE provides numerous procedures, including display procedures, which are callable from the backend. There are two mechanisms for calling these procedures. (see "MAKING a STAND ALONE TOOL" for information on how the Backend obtains the addresses of the relevant SAFE procedures)

The simplest mechanism, which is analogous to SAFE dispatching, is to make a direct procedure call on the SAFE. Two procedures are accessible in this manner:

```
fshow(
```

```
        string REF %string to show user%,
        boolean %TRUE means confirmation required%)
        Returns one result
            TRUE if successful call
    fshowerror(
        string REF, %error string%
        boolean) %TRUE means confirmation required%
        Returns one result
            TRUE if successful call
```

The more powerful mechanism allows abitrary results to be returned by the SAFE procedure. This method requires that the Backend put all the necessary arguments into one L10 list and provide another L10 list for any results that may be generated. The SAFE procedure is called as follows:

```
    fproccall(
        string REF %proc name%,
        list REF, %addr of L10 list of args or 0 if no args%
        list REF) %addr of L10 list to get results%
        Returns one result
            TRUE if successful call / FALSE otherwise
                (the result list will contain the results if any)
```

Examples:

To tell the user that SAFE is a wonderful thing:

```
    fshow( $"SAFE is a wonderful thing", FALSE);
```

To tell the user that the Backend has given up:

```
    fshowerror( $"Backend has given up", TRUE);
```

The second arg TRUE indicates that the user must type CONFIRM before continuing. This is appropriate only for critical messages.

To call the SAFE routine that clears a window:

```
    fproccall( $"clear-window", $arglist, $resultlist);
```

where arglist is an L10 list containing the arguments for the clear window procedure, ie. a window identifier, and resultlist is an empty L10 list.

Provision for Termination

SAFE provides a builtin parsefunction called feterminate. When a grammar calls this parsefunction, SAFE will process the CML TERMINATION rule for the grammar and then do a HALTF.

Type Conversion

Before passing arguments to an execution function, SAFE converts the CML variable values to appropriate L10 values. In addition, L10 results returned by execution functions are conversely converted. The following table indicates the correspondence between CML values and L10.

CML	L10
string	address of the string
integer	value of the integer
true	TRUE
false	FALSE
list	address of the L10 list
null	value of zero
command word	string if integer token is 0, else list(integer token, string)
point	address of L10 list
address	address of L10 list

MAKING a STAND ALONE TOOL (SAT)

Making a Backend

Backend Programming Conventions

The Backend must contain a procedure which will be referred to in this document as bestart. In the process of combining SAFE and the Backend, this routine will be called by SAFE with the addresses of the externally callable SAFE routines:

```
bestart( sfproccall, sfshow, sfshowerror)
```

Bestart must return the address of the backend's procedure selector array. The format of the procedure selector array is:

The first element is a count.

The remainder of the array is a series of count pairs. Each pair consists of the address of a lowercase string containing the name of a procedure as used in the grammar and the address of the procedure itself.

Example: DECLARE procsel = (3, $"xsearch", $xsearch, $"xsort", $xsort, $"xinsert", $xinsert);

IF the backend writer chooses to do Backend dispatching, the array must have a count of one (1) and a single pair made up of the address of the string "dispatcher" and the address of the dispatcher.

Example: DECLARE procsel = (1, $"dispatcher", $xdispatch);

Otherwise, SAFE will do its own dispatching using the Backend provided dispatch table.

Making the Backend Save File

SAFE provides a version of TENLDR, SAFELDR, that has the L10 runtime environment preloaded and the address at which the backend may start loading (170000B). The backend programmer runs this version of SAFELDR and starts loading at or above (numerically greater) 170000B. The Backend programmer should then set the entry vector location to the address of bestart and then save whatever portion of the address space between 170000B and 600000B, inclusive, that she wishes on a file named BE.SAV. (Note that this procedure, i.e. using SAFELDR as opposed to TENLDR, allows the Backend and SAFE to use the same instance of the L10 runtime environment.)

Combining SAFE and the Backend

To combine SAFE and the BE.SAV file previously made using SAFELDR, SAFE must be executed. When SAFE is executed (by typing SAFE.SAV to the EXEC), it will first initialize the L10 environment and itself; then it will load the appropriate grammar (BE.CGR in the connected directory), any parsefunction data (BE.PFD in the connected directory), and any parsefunction code (BE.PFC in the connected directory). All three of these will be loaded below SAFE-END, ie. inside of SAFE.

Next, SAFE will GET BE.SAV (from the connected directory into its own address space) and do an L10 procedure call to the entry vector location, i.e. to bestart. Bestart will called by SAFE and is expected to follow the conventions described above.

Upon return from bestart, SAFE will do what ever processing it desires, then set its entry vector location and do a HALTF. The backend programmer should then save the entire address space on whatever file is desired. The entry vector will be set so that

when the file is started, execution will begin with SAFE processing the CML INITIALIZATION rule for the tool.

< LJOURNAL, 28660.NLS;1, >, 20-SEP-76 18:37 XXX ;;;; .HJOURNAL="JEW
17-SEP-76 17:17 28660"; Title: .H1="Query3 Weekly Status Report";
Author(s): James E. (Jim) White/JEW; Distribution: /JOHN([INFO-ONLY]
) SRI-ARC([INFO-ONLY]) ; Sub-Collections: NIC SRI-ARC; Clerk: JEW;
.IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES; Origin: < WHITE,
1WSTATUS.NLS;3, >, 17-SEP-76 17:16 JEW ;;;;####;

PREFACE

This is one in a series of weekly status reports, requested informally by NELC, describing the status of the Query3 project being funded under RADC Contract No. F30602-76-C-0230. These reports are provided in addition to the monthly status reports to RADC required by the contract.

WEEKLY ACCOMPLISHMENTS

1. Completed coding, off-line checkout, and compilation of the Query3 Data Management System (DMS). Weighing in at 5K words, DMS is approximately 1/3 of the entire Query3 system.
2. Continued to overhaul other modules, and began off-line checkout of the Field Definition module. The overhauled system should be all checkout out off-line and ready for on-line debugging in about a week and a half.
3. Implemented the display (DNLS) versions of the SHOW and FIND commands (which are only implemented for TNLS in Query2).
4. As required by #3 above, coded an NLS sequence generator to run through the members of an arbitrary Query3 "set". This also lays the groundwork for possible later implementation of a host of useful commands, for example, a Query Output-Quickprint-like command.
5. Cleaned up the implementation of the NEAREST and WITHIN options of SHOW and FIND.
6. As fallout from #5, implemented maximum and minimum criteria which may be applied in the FIND command to numeric fields.
7. Implemented a mechanism for allowing the user to disambiguate between two or more platforms with the same name whenever that situation arises in Query.
8. Met with Dave Maynard and Earl Sacerdoti to discuss the Datacomputer database and possible strategies for interfacing Query3 to it.
9. Made a simple modification to Query2 (displaying "COURSE" instead of "BEARING") at the request of John Schill.

TIME CHARGES

Sep 6 - Sep 10 White 28 hours
Sep 13 - Sep 17 White 39 hours
Maynard 2 hours

< CJOURNAL, 28753.NLS;1, >, 14-OCT-76 08:32 XXX ;;;; Title:
Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ;
Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL,
FOO.NLS;3, >, 13-OCT-76 21:58 JBP ;;;;####;

Milestone Input

To respond to the request for new milestone dates my procedure is to find the completion dates of the items that the milestones I am responsible for depend on and to use the latest of those as my start date. 1

19 Initial NLS Installation 2

the following items must be completed before this tasks starts: 2a

1 completed 2a1

3 completed 2a2

16 30 Sep 76 2a3

56 30 Oct 76 2a4

x3 no date given 2a5

therefore the dates are: 2b

to my knowledge item 16 has not been completed and item 56 may be late so the start dates predicted here may be optimistic. 2b1

start date: 1 nov 76 or at the completion of item x3 whichever is later 2b2

complete date: 1 jan 77 or 1 month after completion of item x3 whichever is later 2b3

remarks: depends on 1, 3, 16, 56, x3. 2b4

36 Initial FE Specification Document 3

the following items must be completed before this tasks starts: 3a

none 3a1

therefore the dates are: 3b

start date: 15 Jul 76 3b1

complete date: DONE 23 Oct 76 3b2

remarks: depends on none. 3b3

38 Initial 11 FE 4

the following items must be completed before this tasks starts: 4a

Milestone Input

1	completed	4a1
3	completed	4a2
x3	no date given	4a3
therefore the dates are:		4b
note item x3 has not been completed yet.		4b1
start date: 15 Jun 76		4b2
complete date: 15 Nov 76 or 1 month after the completion of item x3 whichever is later		4b3
remarks: depends on 1, 3, x3.		4b4

40 Final 11 FE 5

the following items must be completed before this tasks starts:		5a
1	completed	5a1
3	completed	5a2
10	15 Feb 77	5a3
11	15 Feb 77	5a4
12	15 Feb 77	5a5
14	no date given	5a6
16	15 Oct 76	5a7
20	1 May 77	5a8
36	completed	5a9
37	15 Oct 76	5a10
38	15 Nov 76	5a11
39	15 Dec 76	5a12
56	30 Oct 76	5a13
x3	no date given	5a14

therefore the dates are: 5b

start date: 1 May 77 or completion of items 14 and x3 whichever is later 5b1

complete date: 1 Jul 77 or 2 months after the completion of items 14 and x3 whichever is later 5b2

remarks: depends on 1, 3, 10, 11, 12, 14, 16, 20, 36, 37, 38, 39, 56, x3. 5b3

48 IOS Implementation on the 11 6

the following items must be completed before this tasks starts: 6a

none 6a1

therefore the dates are: 6b

start date: 1 Jun 76 6b1

complete date: completed 6b2

remarks: depends on none. 6b3

x1 Final NLS Installation 7

the following items must be completed before this tasks starts: 7a

1 completed 7a1

3 completed 7a2

10 15 Feb 77 7a3

12 15 Feb 77 7a4

14 no date given 7a5

16 15 Oct 76 7a6

56 30 oct 76 7a7

therefore the dates are: 7b

start date: 16 feb 77 or completion of item 14 whichever is later 7b1

complete date: 16 apr 77 or 2 months after the completion of item 14 whichever is later 7b2

remarks: depends on 1, 3, 10, 12, 14, 16, 56. 7b3

x2 Final FE Specification Document 8

the following items must be completed before this tasks starts: 8a

10 15 Feb 77 8a1

11 15 Feb 77 8a2

20 1 May 77 8a3

therefore the dates are: 8b

start date: 1 May 77 8b1

complete date: 1 Jun 77 8b2

remarks: depends on 10, 11, 20. 8b3

x3 Foreman Support of Split Tool Direct Connections 9

 ARC Reprints

.Gcr=1; USER INTERFACE SYSTEM FOR A COMPUTER NETWORK MARKETPLACE
.Center= 3; .Gcr=2; Donald Andrews, Charles Irby*, Andrew Poggio, Richard Watson**.GCR; .H1Sw=On; .H2Sw=On;

ABSTRACT
In this paper we describe the need for a consistent user interface across tools in a network environment and the design of a Frontend to meet this need. The goal of the Frontend is to provide a responsive, uniform, terminal-independent user interface that will also reduce communication costs, permit low-cost modification, and accommodate various classes of tools.

INTRODUCTION .Grab=6;

The development of large private and public computer communications networks is creating a new environment with new requirements. This environment will be characterized by a "network marketplace," where a user may choose from a wide range of computer tools and services from wholesalers, retailers, and brokers [1, 2, 3, 4]. This network marketplace poses an exciting challenge to the technical community: How can we provide users with a consistent, responsive interface to all tools and services available through the network to achieve its full resource sharing potential? How do we create a technology framework that allows the smooth introduction of new tools and services, and promotes change, experimentation, and cross-communication between existing tools and services?

The ARPANET has shown the potential and limitations of a heterogeneous network environment [5, 6]. From a terminal connected to an ARPANET computer the user can access tools and services produced by many organizations on a wide variety of vendor hardware. He can, for example, move files from a 16-bit computer to a 36-bit computer, carry on electronic mail dialog with a geographically distributed community (the members of which use many different computers as "home base"), and utilize a diverse collection of interactive and batch tools or data bases.

However, to take full advantage of current ARPANET tools and services the user must overcome many obstacles. He must set up accounts on each system used and receive separate bills. He must learn the idiosyncrasies of the command or control language and the file naming conventions of each operating system and tool used. It is like speaking German to one tool, French to another, and Russian to a third--he must be truly multilingual. Obtaining documentation or assistance is often difficult. Moving the output from one tool or service to another typically requires many steps. .F="Augmentation Research Center (SRI).Split;page .GPN;";

Much technological development will be necessary to provide the smooth interfaces (computer to computer, tool to tool, user to tools and services) required by a mature network environment. For example, an efficient distributed Network Operating System (NOS) properly integrated into vendors' hardware and operating systems must evolve [7]. The many levels of protocols within the ARPANET provide a base for this evolution [8, 9, 10]; issues in NOS development are being explored through paper studies, prototype advanced protocol environments, and whole prototype systems [11, 12, 13, 14].

The most ambitious of the whole prototype systems is the ARPA-Air Force sponsored National Software Works (NSW) [3, 15, 16]. NSW designers recognized that the target hardware and operating system for software under development would not have all the necessary or best tools to aid many phases of software development. Their

response was a network marketplace environment that allows controlled access to a full range of tools with a user environment more coherent than that presently available in the ARPANET.

The NSW system has four components:.PxlFirstShow=Off;

- A Works Manager component that provides resource access control, a common file system distributed among many host file systems and utilizing a consistent file naming convention, and other services [17].
- A Foreman component, executing on each tool-bearing host (i.e., a computer providing tools for the network), that supplies communication and a controlled interface of the local host operating system and tools to the NSW [18].
- The Interprocess Communication Component [19].
- A Frontend component that provides terminal access to the ARPANET, a set of services creating a coherent NSW user interface to the tools and services of the Works Manager, and an environment to decrease the cost of new tool creation..PxlFirstShow=>1;

This paper describes the Frontend system developed at Stanford Research Institute. The Frontend provides a broad spectrum of services on a computer local to the user (Figure 1). (Many other kinds of frontends are possible, such as a "minimum" frontend [20].) Although it was developed for initial use in the NSW environment, the Frontend can be used in other similar environments or as the user interface for any program independent of a network environment. For this reason, the Frontend will be discussed without further reference to the NSW.

Many of the design issues and solutions chosen are of enough technological interest to warrant separate papers. We have limited this paper to a general discussion of the Frontend design and its motivation, leaving much to the reader's intuition and imagination. A more technical presentation is provided in references 21, 22, and 23.

USER INTERFACE SYSTEM FOR A DISTRIBUTED ENVIRONMENT .Grab=6;

The Frontend will evolve to become the user's "intelligent frontend workplace," providing commonly used tools, user interface assistance to network based tools and services, and background intelligent "agents" to perform other tasks [22, 23, 24]. It is a distributed system comprising several components and auxiliary tools (Figure 2). Some of these execute in a computer local to the user, others on network hosts. As hardware technology costs decrease, the best place to run a particular component may change, with a trend toward more user support functions implemented local to the user. Furthermore, frequently used tools such as those for editing and mail handling will be implemented locally.

The Frontend is designed to meet user interface, efficiency, and system requirements. The major goals are listed here, followed by a discussion of how the Frontend meets each requirement.

.PxlFirstShow=Off;

- Provide a responsive, consistent user interface.
- Reduce communication costs and tool-bearing host overhead.
- Provide terminal-independent user and tool interface.
- Allow low-cost user interface modification and experimentation.
- Accommodate various classes of tools.

1) Provide the user with a responsive, consistent command language, feedback environment, and user services across a collection of independently developed tools. .PxlFirstShow=>1;

The Frontend provides the user with a consistent interaction with all tools. He learns one way to specify commands, specify parameters for commands, back up and edit during specification, and obtain help. Prompting and command feedback are presented uniformly across all tools. Of course, the particular commands, syntax, and vocabulary vary with the tool, but the general man/machine dialog framework is consistent. Simplifying the user's tool environment in this way not only increases his facility and productivity, but also encourages him to try new tools and services.

Important to an efficient user interface is quick response during command specification. When using interactive network-based tools, delays may occur at many points for many reasons, some inherent in network performance, others in the tool-bearing host operating systems. To improve response, the Frontend sidesteps some of these delays by moving most of the user interaction onto a low-cost minicomputer or microcomputer closely associated with the user's terminal. (This solution is made especially attractive by the recent rapid advances in performance/cost of minicomputers and microcomputers.)

An improved user interface does not stop here, however. Other goals include asynchronous operations and the support of a number of services across tools. The Frontend provides an asynchronuos operation capability that allows the user, in some instances, to specify succeeding commands while a previously specified command is being executed. Thus the user can continue working during the execution of a lengthy command.

The Frontend also makes possible several kinds of cross-tool (including cross-host) and multiple-tool services. This powerful capability allows the user to execute commands in many different tools as if he were only using one tool. Several kinds of interaction are possible:

Multiple-tool and cross-tool execution. The Frontend allows the user to be concurrently executing or interacting with more than one tool. The display user may divide his screen into separate display windows for each tool and specify command arguments for a tool in one window from information in a second tool window.

A user programming facility. Users can easily create new high level commands, called Command Sequences, from several commands in one or more tools. Because these commands are independent of any particular tool, a user can invoke them at any time from any tool, or from within another Command Sequence. Command Sequences allow arguments to be specified or collected during execution and permit conditional control based on the results of previous commands. The Frontend system also provides facilities to aid in high level specification and debugging of Command Sequences.

"Meta tool" creation. Occasionally a user may need more than one tool to accomplish his task. As the name implies, a meta tool has a single user interface but can cause execution to occur within several hosts where different tool functions exist.

Terminal linking. In the simplest kind of linking, one Frontend contacts another, setting up a two way link that allows users to talk (type) to one another. The next step we

hope to take will be to provide display linking, a more interesting and powerful capability. This will allow two users at display terminals to share a screen while simultaneously carrying on a conversation by telephone or conventional linking. Terminal linking works across hosts for users with different brands of terminals, without cooperation from any tools. .PxlFirstShow=Off;

2) Reduce communication costs and tool-bearing host operating system overhead for interactive tools. .PxlFirstShow=>1;

It is assumed that a well human-engineered interface should not be sacrificed in the effort to reduce communication costs. Packet networks charge for communications, not by distance or speed of the line, but by numbers of packets transmitted. Currently, most commercially available timesharing systems use line-at-a-time communications to improve communication efficiency (each packet consists of a whole line rather than a single character), but these systems cannot provide an adequate user interface. The Frontend solves this problem by allowing character-at-a-time interaction with the user but interpreting input locally on a small processor in or near the terminal. Thus, in most cases, all the interaction with the user necessary to complete a command is performed locally and is then transmitted at one time in a single packet to the tool host. (Packet sizes are around 500-2000 bits or 60 to 250 characters per packet.) This command-at-a-time operation avoids the expensive overhead associated with addressing and error detection and correction for each packet, and the per packet charging algorithms. It also reduces the cost of a process startup or wakeup within the host operating system, since this process must be activated only when a command is executed, rather than at each character or line transmission. .PxlFirstShow=Off;

3) Provide a terminal-independent interface to tools. .PxlFirstShow=>1;

With the Frontend, the tool builder and installer do not have to be explicitly aware of the user's terminal vendor brand. Information about how to handle different terminal types is built into the Frontend. Two virtual terminal types are initially supported, a one dimensional typewriter type (full or half duplex), and a two dimensional display type, supporting screen pointing, multiple windows, and related services. Tools intended for typewriter terminal users may be used via display with command parameters supplied by pointing rather than typing. Meeting this goal not only reduces tool building and installation costs but also allows easy tracking of rapidly advancing terminal technology [23, 25, 26]. .PxlFirstShow=Off;

4) Provide a means for tool builders, installers, man/machine engineers, or psychologists to easily change or experiment with the user interface to the tool. .PxlFirstShow=>1;

Because user interface design principles are not firmly based on controlled experimental data, user interface problems frequently become apparent only after extensive console experience with the target user community. Thus it is important that the user interface specification be easily changeable and that it be as distinct from the rest of the tool as possible to allow low cost experimental setup.

The Frontend provides this capability by allowing the user interface to be specified in a high-level language specially

designed for this purpose, called the Command Meta Language (CML). These specifications are compiled into a data structure called a Grammar that drives a Command Language Interpreter (CLI). The interface designer has only to make simple edits in the CML specification and recompile it to get a new interface--a matter of minutes or hours. Different interfaces to the same functions may be tried, and user interfaces can be easily tailored to reflect cultural differences or jargon peculiar to different user communities. These changes may affect not only the words that are used in the interactions, but the nature of the interaction, the concepts involved, and the kinds of parameters specified by the user.

To further aid interface experimentation, the Frontend will eventually provide a central point to gather user interaction statistics. Command frequencies, error types and frequencies, numbers of accesses to help facilities, and command specification and system responsiveness timing are a few of the measurements possible in the Frontend.

The statistics package will have two components. A measurement package in the Frontend will collect data on command usage by user, user errors, and tool responsiveness. The result will be a data file for each user session. An analysis tool will summarize the data in a variety of ways, as specified by the analysis tool user (e.g., tables and graphs of averages, distributions of various measurements). The analysis tool will also provide an interface to user programs so that users, managers, and/or tool builders can write their own analysis programs. .PXIFirstShow=Off;

5) Accommodate various classes of tools. .PXIFirstShow=>1;

We envision essentially three classes of tools to be made available within a given network marketplace.

Nonintegrated tools. Such tools were designed for use outside the technology framework of the marketplace. They are made available to users with essentially no change to the tool, but at the cost of a different interface scheme and perhaps poor performance.

Partially integrated tools. These tools were developed outside the technology framework of the marketplace and will be included with some modest changes either to the tool or its user interface specification. Tools in this class have the advantage over nonintegrated tools in that they will achieve better performance and/or provide user interfaces that are more consistent with the other marketplace tools.

Fully integrated tools. This class includes tools either specially built or retrofitted to take full advantage of the given technology framework of the network marketplace, its Network Operating System facilities, Frontend user interface services, and so forth.

.PBS;To make it attractive to tool builders or installers to move toward integrated tools and create a coherent user environment, all three tool classes must be accommodated. The Frontend meets this goal by offering a set of services that make it easy, with few or no changes to the tool, either to partially integrate a tool, or to build an integrated tool and take full advantage of the user interface services of the Frontend.

In summary, the Frontend serves as a central coordination point and is with the user at all times. The Frontend knows who the user is

and will adjust certain user interaction characteristics to suit each user by consulting a special data base, called the User Profile. The user can specify which keys perform which command language control functions and how much command feedback and command prompting he is to see. He can define and start up Command Sequences, operations using one or more different commands. The user can actually have several tools working for him at once, though he will interact with only one at a time, and he can use single Grammars that invoke the services of many tools. Statistics on his command usage and his errors can be compiled.

THE STRUCTURE OF THE USER INTERFACE SYSTEM .Grab=6;

To meet our goals we decided that the best approach to the Frontend is a careful modular design and interface specification to create the desired user environment, handle extended capabilities and distributed functions, and provide for future extensions. The major components of the user interface system are shown in Figure 2. They may be separated into three classes: .PxlFirstShow=Off;

- Modules. The programs with which the user interacts during command specification and which communicate with the tool: the Virtual Terminal Control (VTC), the Command Language Interpreter (CLI), and the Process Communication Interface (PCI).
- Data Bases. The data bases and data structures associated with the user interface machinery: the Grammar, User Profile and Help data bases, User Statistics, CML source programs, and Command Sequences.
- Auxiliary Tools. The auxiliary programs and tools that allow the user or tool builder/installer to create, examine, or manipulate the above data bases: the CML Compiler, User Profile tool, Help tool, Statistics Analysis Programs, and Command Sequence Processor. .PxlFirstShow=>1;

Another set of programs associated with the Frontend is not discussed in this paper [27]. These programs control the local peripheral devices such as printers, tape drives, and card equipment that allow information to enter or leave the marketplace.

The user interface system is a distributed system. It is assumed that the VTC, CLI, and PCI will execute on a processor local to the user, although they can be accessed on a processor remote from the user if a local capability does not exist. Grammar and User Profile data bases may be stored remote from the user and shipped to his Frontend processor when needed. It is important to run these processes local to the user--eventually as part of his terminal--to achieve the responsiveness and economies described earlier.

Presently the auxiliary programs and tools run on a host remote from the user's local processor. Eventually we expect some of these, such as the Help tool, to be implemented local to the user.

Because one of our paramount goals is a flexible user interface that can be shaped to the needs of both the user and his tools, it was apparent that the user interface machinery should be driven by data structures. Two main data structures were required--one to represent the user interface syntax of the tool, including feedback, and the second to represent the individual user's choices for various interface services, such as the amount of feedback and prompting he desires. We call the former data structure a Grammar and the latter a User Profile. The same User Profile is referenced throughout a user's session; the Grammar changes each time he uses a different tool.

We have provided a special tool that enables the user to change his User Profile. This points out another design decision: the user interface system actually requires the existence of special auxiliary tools. The User Profile can be changed permanently or temporarily by the tool or by other processes, thus allowing the tool to override User Profile settings or the User Profile to be tied in at a later point with various kinds of adaptive processes.

Help is another auxiliary tool important to the user interface. By typing the HELP key, the user can get information about the use and effects of a command or about the tool in general [28]. When the Help tool is invoked command state information is passed to the tool, enabling it to take the user to an initial point that describes the command and field he was specifying. The Help data base is derived from the grammar syntax and text provided by the tool implementers; it describes in English the intended use of the tool and its commands.

A fundamental design question was how to create the Grammar data structure for a user interface to a given tool that would:

.PxiFirstShow=Off;

- Represent user interfaces for a wide variety of tools.
- Make it very easy to specify and modify the user interface.
- Make the interface easily usable for many people, including nonprogrammers. .PxiFirstShow=>1;

We decided to design a special high-level language for user interface specification, called the Command Meta Language (CML) [21, 23]. A CML program defining a user interface for a tool is compiled by the CML compiler to produce the Grammar, which is interpreted by the CLI. The CML and CLI provide complementary services: the CML supplies the command language specification, embodied in the Grammar; the CLI determines the interaction discipline, which is uniform across tools. A variety of CLIs can be designed with different user interface services.

As a logical extension of the CML decision, tools have been split into two parts--one part to be modeled in the Frontend to handle the user interface, and the other, called the Backend, possessing the information processing facilities in the (possibly remote) host. Some user interface services are essentially tool and context independent, while others depend on the past context of actions and are represented in tool-dependent data structures, or depend on information currently in a display window. These tool-dependent services are supported by the Frontend facilities that allow collaboration between the Frontend and Backend during command specification. The Backend can also send data structures to the Frontend that alter the current Grammar based on context.

Another important design problem was how to handle the three classes of tools described earlier, nonintegrated, partially integrated, and fully integrated tools.

Nonintegrated tools could be handled by simply making the Frontend transparent, i.e., not providing any services to the user to make his interface more consistent. Tools can, in fact, be so installed, but a more effective Frontend can easily be provided. Most existing tools have command languages comprised of fields terminated by some set of characters. If the Frontend buffers characters that are input before a terminating character, some local editing capability can be provided before transmission occurs. This is handled at tool installation time, without tool modification, by specification of a

simple Grammar indicating the transmit character set.
For partially-integrated tools the Frontend allows communication with the tool either as a character string or as a procedure-like call on Frontend basic primitives. To take advantage of some Frontend services (such as the Help features, consistent intracommand editing characters, and the User Profile), the tool installer can easily write a user interface description for his tool in CML. This user interface can be either the same as or quite different from that which previously existed, but in any case the tool itself will not require any modification.
Depending on the size and complexity of the tool's user interface, this task might take a few minutes or a few hours. The result of the successful parse of a command is a string or strings in the old command language. Strings returning from the tool as error messages are parsed and either converted or passed on to the user. If the tool was originally designed to support one-dimensional typewriter terminals and the installer would like to take advantage of two-dimensional displays with pointing devices for argument specification, multiple windows, and random text and picture positioning, minimal additional code at the Frontend can be written to do so without tool modification.
Fully integrated tools utilize all of the Frontend services. To implement a fully integrated tool, the tool builder creates a set of primitives for the information-processing functions of the tool, a communication interface for the network marketplace communication protocol, a description in CML of the desired user interface, and an optional Help data base. He does not have to concern himself with terminal control, writing an interpreter, providing Help, or providing User Profile services. If the tool builder wants to modify the user interface he does not have to rewrite the code; instead he just makes a few edits in his CML source file and recompiles the Grammar. Different user interfaces can be created for study or for use by different individuals or groups with special needs. We have found very significant improvements in tool-building productivity working with this approach. Simple user interfaces can be produced in minutes rather than hours, sophisticated interfaces with large command sets in hours rather than weeks.

CONCLUSION .Grad=6;

A user interface system has been described that provides the kind of consistent, responsive user environment necessary for the maturation of a network marketplace. Currently executing on both the DEC PDP-10 under Tenex, System 20 under Tops 20, and the PDP-11, the system is used in diverse applications such as providing distributed tool access on a large computer network and creating powerful local tools when physically combined with tool backends. Its modular implementation makes it relatively easy to transport to other systems. The system has demonstrated substantial improvements in the productivity of both users and tool builders.
The user interface system opens up many possibilities. The ease with which the user interface can be changed and user statistics gathered makes the system an excellent vehicle for human factors research. Automatic adaptation of both the Grammar and User Profile data structures according to the user's proficiency with a tool is possible. The system also provides an environment for applying advanced technology, such as computer voice recognition, to change the physical nature of the man/machine interface.

ACKNOWLEDGMENTS .Grab=6;
This work was supported by the Defense Advanced Research Projects Agency and Air Force Systems Command's Rome Air Development Center, Griffiss AFB, N.Y. The paper reflects the contribution of many people at SRI's Augmentation Research Center who have explored the user interface principles described here, improved the high level language, and implemented the Frontend design. Special acknowledgment goes to Charles Dornbush who, in 1972, created a Command Meta Language/Command Language Interpreter that served as a testbed and prototype for the current CML and CLI components of the Frontend. We also thank Beverly Boli for her outstanding assistance in organizing and editing the paper.

REFERENCES .Pds; .PxIShow=Off; .PxIFirstShow=Off;

1. Stefferud, E. and J. Hootman, "Structure of the Network Marketplace," EASCON '74 Record, pp. 289-293, October 1974.
2. Stefferud, E., "Economics of Network Delivery of Computer Services," Computer Networks, pp. 53-64, June 1976.
3. Carlson, W. E. and S. D. Crocker, "Impact of Networks on the Software Marketplace," EASCON '74 Record, October 24, 1974, SRI-ARC Catalog Item 24309.
4. Engelbart, D. C., R. W. Watson and J. C. Norton, "The Augmented Knowledge Workshop," AFIPS Proceedings, National Computer Conference, Vol. 42, pp. 9-21, June 1973, SRI-ARC Catalog Item 14724.
5. Roberts, L. G. and B. D. Wessler, "Computer Network Development to Achieve Resource Sharing," AFIPS Proceedings, Spring Joint Computer Conference, pp. 543-549, May 1970.
6. Rosenthal., R., "Network Access Techniques--A Review," AFIPS Proceedings, National Computer Conference, pp. 495-500, June 1976.
7. Watson, R. W., "Some Thoughts on System Design to Facilitate Resource Sharing," Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., November 20, 1973, RFC 592, SRI-ARC Catalog Item 20391.
8. Feinler, E. J. and J. B. Postel, ARPA Protocol Handbook, Network Information Center, Stanford Research Institute, Menlo Park, Ca., April 1976, NIC Catalog Item 7104.
9. Kimbleton, S. R. and R. L. Mandell, Distributed Computation Study, Information Sciences Institute, Marina del Rey, Ca., April 1973.
10. Crocker, S. D. et al., "Function-Oriented Protocols for the ARPA Computer Network," AFIPS Proceedings, Spring Joint Computer Conference, Vol. 40, pp. 271-279, May 1972.
11. White, J. E., "A High-Level Framework for Network-Based Resource Sharing," AFIPS Proceedings, National Computer Conference, pp. 561-570, June 1976, SRI-ARC Catalog Item 27197.
12. White, J. E., Elements of a Distributed Programming System, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., January 6, 1976, SRI-ARC Catalog Item 27250.
13. Sunshine, C. A., "Factors in Interprocess Communication Protocol Efficiency for Computer Networks," AFIPS Proceedings, National Computer Conference, pp. 571-576, June 1976.
14. Kimbleton, S. R. and R. L. Mandell, "A Perspective on Network Operating Systems," AFIPS Proceedings, National Computer Conference, pp. 551-559, June 1976.
15. Balzer, R. et al., Design of a National Software Works, Information Sciences Institute, Marina del Rey, Ca., ISI-RR-73-16, December 20, 1975, SRI-ARC Catalog Item 19208.

16. Balzer, R. et al., The National Software Works, Information Sciences Institute, Marina del Rey, Ca., December 20, 1975, SRI-ARC Catalog Item 24716.
17. Milstein, R. E., Works Manager Procedures, Massachusetts Computer Associates, Wakefield, Mass., April 3, 1975.
18. Shantz, R. E. and R. E. Milstein, The Foreman: Providing the Program Execution Environment for the National Software Works (Preliminary), Bolt, Beranek, and Newman, Boston, Mass. and Massachusetts Computer Associates, Wakefield, Mass., March 31, 1975.
19. NSW Protocol Committee, MSG: The Interprocess Communication Facility for the National Software Works (Preliminary), Bolt, Beranek, and Newman, Boston, Mass. and Massachusetts Computer Associates, Wakefield, Mass., January 23, 1976.
20. Rom, R., Minimum Frontend Specification, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., September 23, 1976, SRI-ARC Catalog Item 28672.
21. Irby, C. H., The Command Meta Language System, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., January 6, 1976, SRI-ARC Catalog Item 27266.
22. Andrews, D. I., B. R. Boli and A. A. Poggio, An Introduction to the NSW Frontend, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., December 1976, SRI-ARC Catalog Item 28743.
23. Andrews, D. I., B. R. Boli and A. A. Poggio, A Guide to the Command Meta Language and Command Language Interpreter, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., December 1976, SRI-ARC Catalog Item 28744.
24. Anderson, R. H. and J. J. Gillogly, "The Rand Intelligent Terminal Agent (RITA) as a Network Access Aid," AFIPS Proceedings, National Computer Conference, pp. 501-509, June, 1976.
25. Irby, C. H., "Display Techniques for Interactive Text Manipulation," AFIPS Proceedings, National Computer Conference, May 1974, November 15, 1973, SRI-ARC Catalog Item 20183.
26. Andrews, D. I., "Line Processor: A Device for Amplification of Display Terminal Capabilities for Text Manipulation," AFIPS Proceedings, National Computer Conference, pp. 257-265, June 1974, SRI-ARC Catalog Item 20184.
27. Kremers, J. H., NSW I/O Station Peripheral Support System, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca., April 18, 1976, SRI-ARC Catalog Item 27899.
28. Lehtman, H. G. et al., Query/Help Software and Data Bases, Knowledge Workshop Development--Reported as of 7/74, Augmentation Research Center, Menlo Park, Ca., December 31, 1975, SRI-ARC Catalog Item 22133.

TITLE PAGE .IgLS; .H1Sw=0; .H2Sw=0; .YBS=0; .YBL=0; .PES;
.GYBS=10;.FSW=0;.SP=C;USER INTERFACE SYSTEM FOR A COMPUTER NETWORK MARKETPLACE.GCR=4;Donald Andrews, Charles Irby, Andrew Poggio, Richard Watson.GCR=8;15 November 76.GCR=17;Augmentation Research Center.GCR;Stanford Research Institute.GCR;Menlo Park, California.GCR=2;(415) 326-6200.GCR=5;

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

Some TENEX-TOPS20 Changes: Notes from a Meeting Held the Second Week of December

See also the text files on ISID in directory <DOCUMENTATION> with names TENEX-TOPS20.blat-DIFFERENCES.

Some TENEX-TOPS20 Changes: Notes from a Meeting Held the Second Week of December

The following are some differences between the PDP-10 TENEX and TOPS-20 operating systems and executives as outlined at a meeting held at ARC last week (second week of December 1976). Note that some of the restrictions currently in effect will be lifted as modifications to TOPS-20 are made. There are also files on ISID in directory <DOCUMENTATION> which have the names TENEX-TOPS20.blat-DIFFERENCES. where "blat" is variously EXEC, FILE, JSYS, etc. Be sure to consult those files as well. 1

1. There is no "terminal type lineprocessor" mode. Instead, one should say "tset" to get the appropriate terminal characteristics. If you type "ter line" by mistake, you will be placed in half-duplex mode and will get into trouble in DNLS. To get out of that trouble, type "ter full" to get into full duplex mode again. 1a

2. You must (for now) type either DNLS or TNLS to get the appropriate version of the system. 1b

3. Journal delivery will continue on ISIC and OFFICE-1. No delivery will take place on ISID for now, though you may send items through SENDMAIL on that system. 1c

4. No archive system is currently available on ISID. However, a version of BSYS will come up soon. 1d

5. No JSYS trap mechanisms are currently available. Code has been written for this feature, but has not been debugged. 1e

6. If you go to an inferior exec (as in the NLS "Goto TENEX" command), you do not QUIT to get back (as in TENEX), but rather POP. Additionally, inferior execs may be invoked at the exec level by giving the PUSH command (rather than the TENEX "EXEC" command). 1f

7. Various terminal characteristic-specifying commands in TENEX must be preceded by the TER command: for example, TER RAISE, not RAISE; TER FULL, not FULL, etc. 1g

8. The length of a scroll "page" is set via the TER PAGE nn command rather than via ther TER SCO nn mm command as in TENEX. A useful feature is the fact that while an item is scrolling, control-S may be used to stop the scroll. In all cases, to resume scrolling, one must type control-Q rather than any character as in TENEX. Control-S may be used to prevent printing at any point; it may be cancelled with a control-Q. 1h

9. Information commands MUST be confirmed with a CR in TOPS-20. In some cases, TENEX would accept other conditions (e.g., typing

of ESC character for name completion) as a confirmation. While, completion will still be carried out with the ESC, an additional confirmation via CR is needed. 1i

10. There is no WHERE IS command. Rather, one should use the SYS command with the desired name as a paramter. E.g., SYS LEHTMAN rather than WHERE IS LEHTMAN. 1j

11. DELETE deletes all versions of a file and RENAME renames all versions of a file unless a specific version number is given. One may use delver to get rid of all but the most recent version(s). 1k

12. Filenames may have the following characters in them: alphanumerics, "$", "-". The version number is separated by a "." Control-V may be used to get special characters into a filename, but that control-V also becomes part of the name: 1l

13. In filenames, "*" may represent a partial field. Thus "ab*fg" will cause recognition of "abcdefg", "abcfg", etc. "%" may be used to specify a single character. Thus "ab%" will get "abc" and "abd" but not "abcd". You cannot find out names which do not exist by playing with recognition mode. You may use more than one of these characters in a field, e.g., "*ab*cd%". 1m

14. Passwords may not be changed by the user. Passwords are not encrypted. 1n

15. There are no ephemerons. Thus, for example, "control-C NETSTAT" will lose your core image. You shold do a PUSH to get a new EXEC before giving such a command if you wish to continue processing the previous job. 1o

16. Rubout, DEL, 177B are used as backspace characters rather than control-H, the ASCII backspace character, or control-A, the TENEX backspace character (at the EXEC level). Many subsystems are written to recognize all three as backspce characters. 1p

17. Control-X does not work consistently at the EXEC. Control-C works well: it is not deferred as in TENEX. To delete a command at the EXEC, type control-C. 1q

18. TOPS-20 save files have the extension "EXE" rather than "SAV". Files brought over from TENEX may still have the extension "SAV. 1r

19. Users have access to "SYSTEM" and "DISK" directories. A single character string may be used to search through several directories for files. DSK: and SYS: have well defined meanings

for the search and may be defined as follows: "DSK: := Xxxxxx" and "SYS: := Xxxxxxx". The defaults are DSK: for DSK: and <SUBSYS>,<SYSTEM>,<IA-SUBSYS> for SYS:. Ken Victor has redefined DSK: to be "DSK:,DSK:*.SAV,<VICTOR>,<VICTOR>*.SAV". (This works since "EXE" has a higher search priority than "SAV".) He redefines "SYS:" to be "DSK:,SYS:". 1s

20. Runfiles may be run atuomatically on every login (to avoid the necessity of typing a set of commands such as TSET and redefinitions of DSK: and SYS:. This is set up by having a file with the name "login.cmd" in your login directory. 1t

21. EOLs are not supported in TOPS-20. Outputting an EOL (37B) to a terminal will actually send the EOL character (which will be echoed as a contol-backarrow ("^_"); it will not be translated to a CR-LF sequence. On input, CR is not translated to an EOL; rather typing a CR actually sends a CR-LF. Runfiles, etc. must have CR-LF in them and not EOL. 1u

22. Disk limits are strictly enforced on ISID. You cannot write if you exceed your disk limits! There are two limits for each user: a permanent limit which may not be exceeded for storage and a working limit which may not be exceeded within a session. When you log out, automatic expunges are done on your directory. 1v

23. A user may set an attribute on the number of versions (generations) of a file which will be left around by using the SET FILE GENERATION-RETENTION-COUNT command. Files with the ";T" do not automatically disappear. (No one seems to know what good they are on TOPS-20.) 1w

24. If the system crashes, a file may be bad if pages are not written out. This is just like the old TENEX days. The load on the system is crummy since there are only two disk packs and no drum! A load average of 5 seems to imply bad response to users. 1x

25. The equivalent of TENEX SAVE is TOPS-20 CSAVE. The equivalent of TENEX SSAVE is TOPS-20 SAVE. 1y

26. GTJFN sometimes works differently. 1z

27. TSET must be specified before trying to get into TNLS. It sets you up as an NVT (among other things.) 1a@

28. The PROTECTION command is SET FILE PROTECTION on TOPS-20. 1aa

29. The new TELNET for TOPS-20 is not complete. 1ab

30. Multiple "GET" in FTP works both ways. lac

31. ISID does incremental and full dumps. A full dump is performed once a week. Between full dumps, incremental dumps are done. After each full dump, the preceding weeks incremental dump tapes are thrown away. At the start of a new month, all the tapes from the preceding month, except the last full dump, are thrown away. lad

32. To "link" to another user give the TOPS-20 "TALK" command, rather than the TENEX "LINK" command. Typing LINK to the exec will run the linking loader for you (not a very pleasant experience). lae

```
'(PROPOSALS) "   & #0
USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-
USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-
USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-ISIC   USC-

GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   G
GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   G
GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   GARLICK   G


$LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $
$LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $
$LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $LLG$SHAREDMIDDLE   $

MONDAY, DECEMBER 27, 1976 18:24:43-PST   MONDAY, DECEMBER 27, 1976 18:24:43-PST
MONDAY, DECEMBER 27, 1976 18:24:43-PST   MONDAY, DECEMBER 27, 1976 18:24:43-PST
MONDAY, DECEMBER 27, 1976 18:24:43-PST   MONDAY, DECEMBER 27, 1976 18:24:43-PST
```

new library subsys and new refence manual

this is the reference manual for use of the librarian subsystem. this document obsoletes <28484,> . It contains some typographical corrections as well as describing new features in the librarian subsystem (which exists as <arcsubsys>library.subsys and <arcsubsys>library.cml at isic and isid). the new features allow access to various dates associated with various files. these new date primitives should make the librarian more useful for subsystems which have some files dependent on the status of other files. the new features are: the 'date' NDACTION; the DATEREF FLAGs; and the following builtin (BF) FLAGs: 'libdate', 'sourcedate', 'reldate', 'inxdate', 'ludate', 'lpdate', 'lxdate', 'lcdate', 'lidate', and 'lndate'.

% SEMANTICS % 1

Any lower case items enclosed in quotes ("lowercase") must appear exactly as shown; however, the item could appear in either upper, lower, or mixed case. Any upper case items enclosed in quotes ("UPPERRCASE") must appear exactly as shown. 1a

LIBRARY-BRANCH = 1b

LF-STATEMENT $(SUB-BRANCH) 1b1

a LIBRARY-BRANCH branch consists of an LF-STATEMENT statement with 0 or more SUB-BRANCH branches underneath it. 1b1a

LF-STATEMENT = 1c

$(PRE-COMMANDS) "LF:" SYSGUIDE-LINK $(POST-COMMANDS) 1c1

a LF-STATEMENT statement consists of optional PRE-COMMANDS followed by the literal text "LF:", followed by a link to the sysguide file for the system, followed by any optional POST-COMMANDS. the PRE-COMMANDS will be processed, then the library will be processed, and finally any POST-COMMANDS will be processed. 1c1a

before processing any SUB-BRANCH the current value of the DACTIONs (and the "sysguide" NDACTION) are saved; these values may be modified by the SUB-BRANCHes; however, when the SUB-BRANCH processing is completed (with the exception of a CB-STATEMENT SUB-BRANCH), the orignal values will be restored. for the purpose of the previous sentence, current value means the value after processing any PRE-COMMANDS. 1c1a1

(the "repeat" NDACTION is set to false prior to processing every SUB-BRANCH). 1c1a1a

SUB-BRANCH = 1d

SF-BRANCH / CB-STATEMENT / RF-STATEMENT
/
NP-STATEMENT 1d1

SF-BRANCH = 1e

SF-STATEMENT SF-PLEX 1e1

a SF-BRANCH consists of a branch having a SF-STATEMENT as the top statement, with a SF-PLEX plex under it. the

SF-PLEX will be generated by the librarian if it doesn't exist. 1e1a

SF-STATEMENT = 1f

$(PRE-COMMANDS) "SF:" SOURCE-FILE-LINK $(POST-COMMANDS) 1f1

a SF-STATEMENT statement consists of optional PRE-COMMANDS followed by the literal text "SF:", followed by a link to a source code file or a documentation file, etc., followed by any optional POST-COMMANDS. the PRE-COMMANDS will be processed, then the branch will be processed, (i.e. the source file will be updated, and/or printed, and/or compiled, etc. according to the values of the ACTIONs specified by the PRE-COMMANDs and the values set up by the LF-STATEMENT PRE-COMMANDS) and finally any POST-COMMANDS will be processed. 1f1a

SF-PLEX = 1g

PRH-STATEMENT UPH-STATEMENT PNH-STATEMENT INX-STATEMENT INC-STATEMENT CON-STATEMENT CMP-STATEMENT 1g1

the SF-PLEX is a plex underneatn the SF-STATEMENT consisting of the above named statements. this plex is used to maintain last processed dates, pointers to files, etc. this plex will be created by the librarian if it doesn': exist. as each SF-BRANCH is processed, each statement in the SF-PLEX is copied underneath itself if the associated action is performed, e.g. if the source file is updated, the UPH-STATEMENT will be copied under the UPH-STAIMENT, and tne the appropiate statement is edited to indicate who and when the action was performed. 1g1a

PRH-STATEMENT = 1h

this is the processing history statement for the source file; each time the librarian processes this source file, the LF-STATEMENT for the source file will be copied to underneath this PRH-STATEMENT. 1h1

UPH-STAIEMENT = 1i

this is the update history statement for the source file; each time the librarian updates this source file, the UPH-STATEMENT for the source file will be copied to underneath this UPH-STATEMENT. 1i1

PNH-STATEMENT = 1j

this is the print history statement for the source file; each time the librarian prints this source file, the PNH-STATEMENT for the source file will be copied to underneath this PNH-STATEMENT. 1j1

INX-STATEMENT = 1k

this statement contains a link to the index file for this source file and it is also the index history statement for the source file; each time the librarian indexes this source file, the INX-STATEMENT for the source file will be copied to underneath this INX-STATEMENT. 1k1

INC-STATEMENT = 1l

this statement contains the count of the INCLUDES found in the source file and it is the include count history statement for the source file; each time the librarian counts the INCLUDES in this source file, the INC-STATEMENT for the source file will be copied to underneath this INC-STATEMENT and the count in the INC-STATEMENT will be updated to reflect the most recent counting. the include count in this statement is used to determine whether the source file should be include compiled or normal compiled. 1l1

CON-STATEMENT = 1m

this statement contains the count of the conditional compilation flags found in the source file and it is the conditional count history statement for the source file; each time the librarian counts the conditionals in this source file, the CON-STATEMENT for the source file will be copied to underneath this CON-STATEMENT and the count in the CON-STATEMENT will be updated to reflect the most recent counting. 1m1

CMP-STATEMENT = 1n

this statement contains a link to the branch at which compilation of the source file is to begin, followed by a link to the compiler to be used, followed by a link to the rel file to be generated. it is also is the compile history statement for the source file; each time the librarian compiles this source file, the CMP-STATEMENT for the source file will be copied to underneath this CMP-STATEMENT. 1n1

CB-STATEMENT = 1o

"CB:" $(COMMAND) 1o1

the COMMANDs specified in a CB-STATEMENT remain in effect for any SUB-BRANCHees which follow this branch, i.e. they are similar to specifying PRE-COMMANDS in the LF-STATEMENT. 1o1a

RF-STATEMENT = 1p

$(PRE-COMMANDS) "RF:" SAVE-FILE-LINK INPUT-FILE-LINK $(POST-COMMANDS) 1p1

a RF-STATEMENT statement consists of optional PRE-COMMANDS followed by the literal text "RF:", followed by a link to a save file to be executed, followed by a link to a file to be used as a primary input file for the executuion of the save file, followed by any optional POST-COMMANDS. the PRE-COMMANDS will be processed, then the branch will be processed, (i.e. the save file will be executed) and finally any POST-COMMANDS will be processed. 1p1a

NP-STATEMENT = 1q

$(PRE-COMMANDS) "NP:" SAVE-FILE-LINK INPUT-FILE-LINK $(POST-COMMANDS) 1q1

the only difference between an NP-STATEMENT and an RF-STATEMENT is that for an NP-STATEMENT the succesfull execution of an NP save file is determined by the contents of register one after completion of the inferior fork. (if the NP file aborts, e.g. illegal instruction trap, then the save file will have been considered to not run successfully.) for a RF save file successful execution is determined soley by whether or not the inferior aborts or not. 1q1a

SYSGUIDE-LINK = a link to the sysguide file for the system 1r

SOURCE-FILE-LINK = 1s

a link to a source code file, or a documentation file, etc. for the system 1s1

SAVE-FILE-LINK = a link to a save file to execute 1t

INPUT-FILE-LINK = 1u

a link to a file to be used as the primary input file while executing a SAVE-FILE-LINK file 1u1

PRE-COMMANDS = COMMAND 1v

POST-COMMANDS = COMMAND 1w

COMMAND = 1x

PCMD / TCMD 1x1

COMMANDs are either permanent or temporary. 1x1a

PCMD = 1y

"<<" CMDBODY ">>" 1y1

permanent COMMANDs are enclosed in double angle brackets and are not deleted by the librarian. 1y1a

TCMD = 1z

"##" CMDBODY "##" 1z1

temporary COMMANDs are enclosed in double poundsigns and are processed once by the librarian and then deleted. 1z1a

CMDBODY = 1a@

QUALCLAUSE ACTION 1a@1

the QUALCLAUSE is evaluated according to a 3-value (true, false, maybe) logic system and then the ACTION may or may not be performed depending and the value of QUALCLAUSE. 1a@1a

QUALCLAUSE = 1aa

QUALIFIER $(CEMENT QUALIFER) 1aa1

a QUALCLAUSE consists of a QUALIFIER optionally followed by one or more CEMENT - QUALIFIER pairs. there is no hierarchy of the CEMENT and the QUALCLAUSE is evaluated in a strictly left to right order. each QUALIFIER is evaluated (to true, false, or maybe) and then CEMENTed to the next QUALIFIER according to the intervening CEMENT (according to a 3-value logic system). the value of the QUALCLAUSE is the value when all QUALIFIERs have been CEMENTed together. 1aa1a

QUALIFIER = 1ab

NEVER / 1ab1

this QUALIFIER has the value false. 1ab1a

DEFAULT / 1ab2

this QUALIFIER has the value maybe. 1ab2a

ALWAYS / 1ab3

this QUALIFIER has the value true. 1ab3a

IFF FLAG / 1ab4

this QUALIFIER has the value true if FLAG is false;
otherwise it has the value false. 1ab4a

IFM FLAG / 1ab5

this QUALIFIER has the value true if FLAG is maybe;
otherwise it has the value false. 1ab5a

IFT FLAG / 1ab6

this QUALIFIER has the value true if FLAG is true; otherwise
it has the value false. 1ab6a

IFNF FLAG / 1ab7

this QUALIFIER has the value true if FLAG is true or maybe;
otherwise it has the value false. 1ab7a

IFNM FLAG / 1ab8

this QUALIFIER has the value true if FLAG is true or false;
otherwise it has the value false. 1ab8a

IFNT FLAG / 1ab9

this QUALIFIER has the value true if FLAG is false or maybe;
otherwise it has the value false. 1ab9a

IFE FLAG (FLAG / number) / 1ab10

this QUALIFIER is true if the numerical value of the first
FLAG is the same as the numerical value of the second FLAG
or number; otherwise it has the value false. 1ab10a

IFNE FLAG (FLAG / number) / 1ab11

this QUALIFIER is true if the numerical value of the first
FLAG is not the same as the numerical value of the second
FLAG or number; otherwise it has the value false. 1ab11a

TFG FLAG (FLAG / number) / 1ab12

this QUALIFIER is true if the numerical value of the first FLAG is greater than the numerical value of the second FLAG or number; otherwise it has the value false. 1ab12a

TFNG FLAG (FLAG / number) / 1ab13

this QUALIFIER is true if the numerical value of the first FLAG is not greater than the numerical value of the second FLAG or number; otherwise it has the value false. 1ab13a

TFL FLAG (FLAG / number) / 1ab14

this QUALIFIER is true if the numerical value of the first FLAG is less than the numerical value of the second FLAG or number; otherwise it has the value false. 1ab14a

TFNL FLAG (FLAG / number) 1ab15

this QUALIFIER is true if the numerical value of the first FLAG is not less than the numerical value of the second FLAG or number; otherwise it has the value false. 1ab15a

CEMENT = 1ac

AND / OR / XOR 1ac1

CEMENT ands, ors, or exclusive ors the QUALIFIERs on both sides according to the 3-value logic system discussed below. 1ac1a

ACTION = 1ad

DACTION / NDACTION / FACTION 1ad1

an ACTION is performed depending on the type of ACTION it is and the resulting value of QUALCLAUSE. for an FACTION, the value of QUALCLAUSE is stored in the specified FLAG; an NDACTION is performed if QUALCLAUSE has a value of true or maybe, otherwise it is not performed; a DACTION is performed if QUALCLAUSE has the value true, not performed if QUALCLAUSE has the value false, and if QUALCLAUSE has the value maybe the the DACTION will cause two dates (e.g. the date of the source code and the date of the rel file) to be compared and if the first date is more recent than the second date, the DACTION will be performed. (see DACTION below for which dates are compared). 1ad1a

DACTION = 1ae

1ae1

for the following DACTIONs, if QUALCLAUSE has the value maybe then the date of the source file is compared with the date of appropriate other dates. the date of the source file (source date) is the write date for the source file if the file does not have a PC, or the maximum positive number if the file has a PC. 1ae1a

"everything" / 1ae2

"everything" is a shorthand way for specifying all the rest of the DACTION actions and the "sysguide" NDACTION. 1ae2a

"update" ["new"/"compact"/"old"] / 1ae3

dates compared: source date : maximum positive number - 1 1ae3a

the absence of a modifier ("new"/"old"/"compact") is equivalent to a modifier of "old". 1ae3b

"print" / 1ae4

dates compared: source date : PRINT-DATE 1ae4a

if the file is printed, it will be printed using a special content analyzer pattern. PRINT-DATE is the date in the PNH-STATEMENT statement for the source file. 1ae4b

"index" / 1ae5

dates compared: source date : INDEX-DATE 1ae5a

INDEX-DATE is the write date for the index file associated with the source file. 1ae5b

"include" / 1ae6

dates compared: source date : INCLUDE-DATE 1ae6a

INCLUDE-DATE is the date in the INC-STATEMENT statement for the source file. 1ae6b

"conditional" / 1ae7

dates compared: source date : CONDITIONAL-DATE 1ae7a

CONDITIONAL-DATE is the date in the CON-STATEMENT statement for the source file. 1ae7b

"compile" 1ae8

dates compared: source date : COMPILE-DATE 1ae8a

COMPILE-DATE is the write date for the rel file. 1ae8b

NDACTION = 1af

"sysguide" / 1af1

for all files for which an index file exists or is generated and for which the QUALCLAUSE for "sysguide" is true or maybe, the index file is copied into the sysguide file. 1af1a

"detach" / 1af2

the librarian will detach the job if this NDACTION is performed. this NDACTION is only checked in the LF-STATEMENT 1af2a

"abort" / 1af3

library processing will be aborted if this NDACTION is performed. 1af3a

"output" ["to"] TENEX-FILE-NAME / 1af4

if this NDACTION is performed, then any output generated by the librarian will be written on the specified file; note that tenex format is used to specify the file and not link syntax; the TENEX-FILE-NAME should be terminated byone or more invisibles. 1af4a

"runfile" / 1af5

if this NDACTION is performed (i.e. its associated QUALCLAUSE has the value true or maybe), then inferior forks (as specified by RF-STATEMENTS or NP-STATEMENTS) will be executed. 1af5a

"ignore" / 1af6

if this NDACTION is performed, then the branch in which it occurs will be ignored. 1af6a

"status" FLAG / 1af7

```
      if this NDACTION is performed, then the logical value of the
      named flag will be written on the output file.                  1af7a

   "decimal" FLAG /                                                   1af8

      if this NDACTION is performed, then the decimal value of the
      named flag will be written on the output file.                  1af8a

   "octal" FLAG /                                                     1af9

      if this NDACTION is performed, then the octal value of the
      named flag will be written on the output file.                  1af9a

   "date" FLAG /                                                      1af10

      if this NDACTION is performed, then the value of the named
      flag will be written on the output file as a time and date.     1af10a

   "comment" COMMENT /                                                1af11

      if this NDACTION is performed, then the COMMENT text will be
      written on the output file.                                     1af11a

   "logout" /                                                         1af12

      if this NDACTION is performed, then the job will be logged
      out after the LIBRARIAN finishes processing.  This NDACTION
      is only checked in the LF-STATEMENT.                            1af12a

   "wait" ("for" [HH:]MM / "until" [HH:]MM) /                         1af13

      if this NDACTION is performed, the LIBRARIAN will wait
      either for the specified number of (optional hours and)
      minutes, or until the specified (optional hour and) minute,
      to start processing.  If a "wait until" is specified, then
      HH if specified is based on a 24-hour clock; if HH is not
      specified then it is assumed to be 00, i.e. midnight.  This
      NDACTION is only checked as a PRE-COMMAND in the
      LF-STATEMENT.                                                   1af13a

   UFOP FLAG /                                                        1af14

      if this NDACTION is performed, then the appropriate UFOP is
      performed on the specified flag.                                1af14a

   BFOP FLAG FLAG /                                                   1af15

      if this NDACTION is performed, then the appropriate BFOP is
      performed on the specified flags.                               1af15a
```

"repeat" 1af16

if this NDACTION is performed, then the branch in which the COMMAND occurs will be repeated. 1af16a

FACTION = 1ag

FLAG 1ag1

the value of QUALCLAUSE is stored into the specified FLAG. 1ag1a

UFOP = 1ah

FSETF / 1ah1

this unary flag operator will set the logical value of the named flag to false. 1ah1a

FSETM / 1ah2

this unary flag operator will set the logical value of the named flag to maybe. 1ah2a

FSETT / 1ah3

this unary flag operator will set the logical value of the named flag to true. 1ah3a

FNOT 1ah4

this unary flag operator will set the logical value of the named flag to its logical not. 1ah4a

BFOP = 1ai

FOR / 1ai1

this binary flag operator will set the logical value of the first named flag to the logical or of the first and second named flags. 1ai1a

FAND / 1ai2

this binary flag operator will set the logical value of the first named flag to the logical and of the first and second named flags. 1ai2a

FXOR / 1ai3

this binary flag operator will set the logical value of the first named flag to the logical exclusive or of the first and second named flags. 1ai3a

FGET / 1ai4

this binary flag operator will set the numerical value of the first named flag to the numerical of the second named flag. the second named flag is allowed to be an (optionally signed) number as opposed to a FLAG. 1ai4a

FADD / 1ai5

this binary flag operator will set the numerical value of the first named flag to the numerical sum of the first and second named flags. the second named flag is allowed to be an (optionally signed) number as opposed to a FLAG. 1ai5a

FSUB 1ai6

this binary flag operator will set the numerical value of the first named flag to the numerical difference of the first named flag minus the second named flag. the second named flag is allowed to be an (optionally signed) number as opposed to a FLAG. 1ai6a

FLAG = 1aj

UF / BF / DATEREF 1aj1

there are 3 types of flags: user settable flags (UFs); builtin flags (BFs); and date references (DATEREFs). 1aj1a

UF = "uf" followed immediately by a number from 1 to 40, e.g. uf1 1ak

1ak1

user settable flags are initialized to false. they can be set and tested with the appropriate commands. they are not reset at the start of each new SB-BRANCH. 1ak1a

DATEREF = 1al

1al1

Note that in the following 3 FLAGs, a TENEX file name is used and not link syntax; also note that the specified TENEX file name MUST be terminated by one or more invisibles. Note also that if the specified file refers to an NLS file,

NO checks will be made to see if the file has a partial copy, and that the time and date of the named file will be used, NOT the time and date of the partial copy (if one exists). 1a11a

"writedate" TENEX-FILE-NAME / 1a12

This DATEREF has the value of the time and date that the specified file was last written. 1a12a

"readdate" TENEX-FILE-NAME / 1a13

This DATEREF has the value of the time and date that the specified file was last read; if the file has not been read, it will have the value zero (0). 1a13a

"createdate" TENEX-FILE-NAME 1a14

This DATEREF has the value of the time and date that the specified file was created. 1a14a

BF = 1am

1am1

BFs are of various types: some are reset to either an initial or computed value prior to the processing of every SUB-BRANCH; some are reset to either an initial or computed value at the start of processing of certain types of SUB-BRANCHs; some are set during the processing of SUB-BRANCHes; and some are running counters. 1am1a

"libdate" / 1am2

when reset: start of execution of the librarian 1am2a

reset value: 1am2b

the time and date the librarian was last executed, i.e., the time and date from the LF-STATEMENT. 1am2b1

when set: never 1am2c

set value: none 1am2d

"sourcedate" / 1am3

when reset: start of processing of SF-BRANCHes 1am3a

reset value: 1am3b

the time and date the file named by the SOURCE-FILE-LINK in the SF-STATEMENT was last written; if the file has a patial copy, the value of this flag will be the time and date of the partial copy, i.e., the current time and date since the librarian will have opend the partial copy by the time it sets this flag. 1am3b1

when set: never 1am3c

set value: none 1am3d

"reldate" / 1am4

when reset: start of processing of SF-BRANCHes 1am4a

reset value: 1am4b

the time and date the file named as the output file in the CMP-STATEMENT was last written (i.e., the write date of the rel file named in the third link in the CMP-STATEMENT). 1am4b1

when set: never 1am4c

set value: none 1am4d

"inxdate" / 1am5

when reset: start of processing of SF-BRANCHes 1am5a

reset value: 1am5b

the time and date the file named as the index file in the INX-STATEMENT was last written; if for some reason the librarian can't load the index file, this falg will have the value 0; if the file has a partial copy, this flag will have the value 35M, i.e. the maximum positive number. 1am5b1

when set: never 1am5c

set value: none 1am5d

"ludate" / 1am6

when reset: start of processing of SF-BRANCHes 1am6a

reset value: 1am6b

the time and date the librarian last updated the file named in the SF-STATEMENT, i.e., the time and date from the UPH-STATEMENT. 1am6b1

when set: never 1am6c

set value: none 1am6d

"lpdate" / 1am7

when reset: start of processing of SF-BRANCHes 1am7a

reset value: 1am7b

the time and date the librarian last printed the file named in the SF-STATEMENT, i.e., the time and date from the PNH-STATEMENT. 1am7b1

when set: never 1am7c

set value: none 1am7d

"lxdate" / 1am8

when reset: start of processing of SF-BRANCHes 1am8a

reset value: 1am8b

the time and date the librarian last indexed the file named in the SF-STATEMENT, i.e., the time and date from the INX-STATEMENT. 1am8b1

when set: never 1am8c

set value: none 1am8d

"lcdate" / 1am9

when reset: start of processing of SF-BRANCHes 1am9a

reset value: 1am9b

the time and date the librarian last compiled the file named in the SF-STATEMENT, i.e., the time and date from the CMP-STATEMENT. 1am9b1

when set: never 1am9c

set value: none 1am9d

"lidate" / 1am10

when reset: start of processing of SF-BRANCHes 1am10a

reset value: 1am10b

the time and date the librarian last counted the INCLUDE statements the file named in the SF-STATEMENT, i.e., the time and date from the INC-STATEMENT. 1am10b1

when set: never 1am10c

set value: none 1am10d

"lndate" / 1am11

when reset: start of processing of SF-BRANCHes 1am11a

reset value: 1am11b

the time and date the librarian last counted the conditional compile statements the file named in the SF-STATEMENT, i.e., the time and date from the CON-STATEMENT. 1am11b1

when set: never 1am11c

set value: none 1am11d

"filncmp" / 1am12

when reset: start of processing of SF-BRANCHes 1am12a

reset value: true 1am12b

when set: after the SF-BRANCH file is compiled or not compiled 1am12c

set value: true if the SF-BRANCH file is not compiled 1am12d

"filcmp" / 1am13

when reset: start of processing of SF-BRANCHes 1am13a

reset value: false 1am13b

when set: after the SF-BRANCH file is compiled or not compiled lam13c

set value: true if the SF-BRANCH file is compiled lam13d

"oldfile" / lam14

when reset: start of processing of SF-BRANCHes lam14a

reset value: true if rel file more recent than source file; else false lam14b

when set: never lam14c

set value: none lam14d

"newfile" / lam15

when reset: start of processing of SF-BRANCHes lam15a

reset value: true if source file more recent than rel file; else false lam15b

when set: never lam15c

set value: none lam15d

"cmpok" / lam16

when reset: start of processing of SF-BRANCHes lam16a

reset value: true lam16b

when set: after the SF-BRANCH file is compiled or not compiled lam16c

set value: true if file compiled without errors; else false lam16d

"cmperr" / lam17

when reset: start of processing of SF-BRANCHes lam17a

reset value: false lam17b

when set: after the SF-BRANCH file is compiled or not compiled lam17c

set value: true if file compiled with errors; else false lam17d

```
"noerrors" /                                                    lam18
   when reset: start of library processing                      lam18a
   reset value: true                                            lam18b
   when set: after any file is compiles with errors             lam18c
   set value: false                                             lam18d
"totalerrors" /                                                 lam19
   when reset: start of library processing                      lam19a
   reset value: 0                                               lam19b
   when set: after any file is compiles with errors             lam19c
   set value: "totalerrors" + 1                                 lam19d
"filepc" /                                                      lam20
   when reset: start of processing of SF-BRANCHes               lam20a
   reset value:                                                 lam20b
      true if source file has a PC to which the librarian has
      access; else false                                        lam20b1
   when set: never                                              lam20c
   set value: none                                              lam20d
"nopc" /                                                        lam21
   when reset: start of processing of SF-BRANCHes               lam21a
   reset value: logical not of "filepc"                         lam21b
   when set: never                                              lam21c
   set value: none                                              lam21d
"infrun" /                                                      lam22
   when reset: start of RF-STATEMENT or NP-STATEMENT processing lam22a
   reset value: false                                           lam22b
```

when set: after an inferior fork is run 1am22c

set value: true 1am22d

"infnrun" / 1am23

when reset: start of RF-STATEMENT or NP-STATEMENT processing 1am23a

reset value: true 1am23b

when set: after an inferior fork is run 1am23c

set value: false 1am23d

"infok" / 1am24

when reset: start of RF-STATEMENT or NP-STATEMENT processing 1am24a

reset value: true 1am24b

when set: after an inferior fork is run 1am24c

set value: 1am24d

for an RF-STATEMENT, set to true if the inferior terminates normally, other false; for an NP-STATEMENT set to the contents of register one if the inferior terminates normally, otherwise set to false. 1am24d1

"infnok" / 1am25

when reset: start of RF-STATEMENT or NP-STATEMENT processing 1am25a

reset value: true 1am25b

when set: after an inferior fork is run 1am25c

set value: logical not of "infok" 1am25d

"numcompiles" / 1am26

when reset: start of library processing 1am26a

reset value: 0 1am26b

when set: after any file is compiled 1am26c

set value: "numcompiles" + 1 1am26d

```
    "numokcompiles" /                                                   lam27
       when reset: start of library processing                          lam27a
       reset value: 0                                                   lam27b
       when set: after any file is compiled without any errors          lam27c
       set value: "numokcompiles" + 1                                   lam27d
    "numicompiles" /                                                    lam28
       when reset: start of library processing                          lam28a
       reset value: 0                                                   lam28b
       when set: after any file is "INCLUDE" compiled                   lam28c
       set value: "numicompiles" + 1                                    lam28d
    "numrcompiles" /                                                    lam29
       when reset: start of library processing                          lam29a
       reset value: 0                                                   lam29b
       when set: after any file is "REGULAR" compiled                   lam29c
       set value: "numrcompiles" + 1                                    lam29d
    "numprints" /                                                       lam30
       when reset: start of library processing                          lam30a
       reset value: 0                                                   lam30b
       when set: after any source file is printed                       lam30c
       set value: "numprints" + 1                                       lam30d
    "numupdates" /                                                      lam31
       when reset: start of library processing                          lam31a
       reset value: 0                                                   lam31b
       when set: after any source file is updated                       lam31c
       set value: "numupdates" + 1                                      lam31d
```

```
"numindexes" /                                                  1am32
   when reset: start of library processing                      1am32a
   reset value: 0                                               1am32b
   when set: after any source file is indexed                   1am32c
   set value: "numindexes" + 1                                  1am32d
"numsysguides" /                                                1am33
   when reset: start of library processing                      1am33a
   reset value: 0                                               1am33b
   when set: after any source file index is copied to the
   system sysguide                                              1am33c
   set value: "numsysguides" + 1                                1am33d
"numincludes" /                                                 1am34
   when reset: start of library processing                      1am34a
   reset value: 0                                               1am34b
   when set: after any source file is include counted           1am34c
   set value: "numincludes" + 1                                 1am34d
"numconditionals" /                                             1am35
   when reset: start of library processing                      1am35a
   reset value: 0                                               1am35b
   when set: after any source file is conditional counted       1am35c
   set value: "numconditionals" + 1                             1am35d
"numinferiors" /                                                1am36
   when reset: start of library processing                      1am36a
   reset value: 0                                               1am36b
   when set: after any inferior is run                          1am36c
```

```
        set value: "numinferiors" + 1                                   1am36d

    "numokinferiors"                                                    1am37

        when reset: start of library processing                         1am37a

        reset value: 0                                                  1am37b

        when set: after any inferior is run with a completion of
        true or maybe                                                   1am37c

        set value: "numokinferiors" + 1                                 1am37d

FOR = "for"                                                             1an

FAND = "fand"                                                           1ao

FXOR = "fxor"                                                           1ap

FGET = "fget"                                                           1aq

FADD = "fadd"                                                           1ar

FSUB = "fsub"                                                           1as

FSETF = "fsetf"                                                         1at

FSETM = "fsetm"                                                         1au

FSETT = "fsett"                                                         1av

FNOT = "fnot"                                                           1aw

HH = a number; (for "wait until" must be in range of 0 to and
including 23)                                                           1ax

MM = a number; (for "wait until" must be in range of 0 to and
including 59)                                                           1ay

AND = ".a"                                                              1az

OR = ".o"                                                               1b@

XOR = ".x"                                                              1ba

NEVER = "never"                                                         1bb

DEFAULT = "default"                                                     1bc
```

```
    ALWAYS = "always"                                                   1bd
    TFF = "tff"                                                         1be
    TFM = "tfm"                                                         1bf
    TFT = "tft"                                                         1bg
    TFNF = "tfnf"                                                       1bh
    TFNM = "tfnm"                                                       1bi
    TFNT = "tfnt"                                                       1bj
    TFE = "tfe"                                                         1bk
    TFNE = "tfne"                                                       1bl
    TFG = "tfg"                                                         1bm
    TFNG = "tfng"                                                       1bn
    TFL = "tfl"                                                         1bo
    TFNL = "tfnl"                                                       1bp

                                                                        1bq
Discussion of 3-value logic system                                      2
    true is any flag with a numerical value greater than 0;             2a
    maybe is any flag with a numerical value equal to 0;                2b
    false is any flag with a numerical value less than 0.               2c
    the following are the truth tables used:                            2d
      OR                                                                2d1
                    FALSE   MAYBE   TRUE                                2d1a
         FALSE      false   maybe   true                                2d1b
         MAYBE      maybe   maybe   true                                2d1c
          TRUE      true    true    true                                2d1d
      AND                                                               2d2
```

	FALSE	MAYBE	TRUE	
	FALSE	MAYBE	TRUE	2d2a
FALSE	false	false	false	2d2b
MAYBE	false	maybe	maybe	2d2c
TRUE	false	maybe	true	2d2d

XOR 2d3

	FALSE	MAYBE	TRUE	
	FALSE	MAYBE	TRUE	2d3a
FALSE	false	maybe	true	2d3b
MAYBE	maybe	maybe	maybe	2d3c
TRUE	true	maybe	false	2d3d

NOT 2d4

FALSE	MAYBE	TRUE	
			2d4a
true	maybe	false	2d4b

AN INTERACTIVE WORKSTATION FOR KNOWLEDGE WORKERS

Robert Louis Belleville
Stanford Research Institute
Menlo Park, California 94025

Abstract

Design considerations for the development of a low cost, high performance, interactive, mixed text and graphics workstation are described in this paper. The use of video technology is discussed.

1. INTRODUCTION

The engineer requires access to a variety of computer systems to effectively function in an on-line environment. Among these are:

1) Access to numerous data bases and analytical tools.

2) Communication with co-worker and management.

3) Ability to manipulate textual and graphical entities such as:

   a) specifications and requirements,

   b) design notes,

   c) cost and scheduling data,

   d) documentation.

The computer terminal is becoming the engineer's gateway to a large number of systems to assist him in his "knowledge work." A single, high performance terminal is needed to allow the engineer uniform access to these tools. 4b

Specifications for such a terminal include: 4c

1) The ability to display a full typewritten page of text including special symbols. A pica typewritter can put 66 lines of 85 characters on a 8.5 by 11.0 inch page. Typical margins reduce this figure to 54 lines of 72 characters. 4c1

2) The ability to create and display drawings. 4c2

3) Interactive devices for typing and pointing. 4c3

4) Conformance with the office environment. This includes adequate workspace, user comfort, and quiet. 4c4

To produce a full page of text with mixed graphics at moderated cost, requires the use of raster-scan or video displays. 4d

2. VIDEO DISPLAYS 5

2.1 ADVANTAGES OF VIDEO 5a

Raster-scan displays have several advantages. 5a1

First, video displays are low cost. Complex parts such as the CRT itself, which would be prohibitively expensive to produce in small quantities, are cheap and plentiful because of the impact of broadcast tv. Moreover, the technology is well worked out, standards have been adopted, and many manufacturers compete in an open market. While the home tv has relatively low picture quality, broadcast studios and the closed circuit tv market have produced a demand for very high quality tv equipment at reasonable cost. 5a2

Second, the video waveform is attractive for several reasons: 5a3

1) The image is coded in a standardized way. 5a3a

2) This code provides for the creation of an image from a single stream of data on a single wire. No complex interface is required between the source of the video signal and the tv monitor. Part of the intelligence needed to describe the picture is contained in the monitor itself. 5a3b

All the information to construct the picture is contained in the low energy video channel. Random or calligraphic displays distribute the information into 2 high energy position channels and a low energy intensity channel. Video on the other hand uses two positioning channels which operate

at fixed frequencies so that the driving circuitry can be optimized to minimize the complexity and power requirements. 5a3b1

3) The code provides for the creation of not only line drawings but also pictures containing grey levels or color. In its simplest form, computer generated video need only contain a binary level - on or off; but if necessary, the computer video generator can produce grey levels and color. 5a3c

4) Although there is no logical information in the video signal, the waveform is rich in spatial information. This information can be used to mix video waveforms from several sources into a single video output, and to provide split screen and overlay characteristics. 5a3d

5) The bandwidth of the source and the monitor will control the density of information displayed to the user. The information will be displayed without flicker or drift. 5a3e

Third, video technology is based on memory. No matter which of the several video production techniques are used, the designer must find relatively large quantities of memory. Video is thriving today because memory cost is dropping dramatically, memory speed is increasing, memory density is increasing, and the power consumption is dropping. In short, the vital element needed for video production is becoming the simplest and cheapest part of the system. 5a4

Fourth, related technologies are advancing. 5a5

1) The creation of the data structures needed to convert pictures described as computer based data structures (both lines and surfaces) is relatively well understood. The process of scan conversion requires considerable processor speed, but here again, the speed of processors is increasing while the price drops. 5a5a

2) Electrostatic printer/plotters are both fast and reliable. And the price is quite reasonable. The technology needed to drive the electrostatic printer is identical to that needed for the production of video. 5a5b

3) The use and storage of pictures, that is photographs, is increasing and the technology for the encoding of these images relates closely to the problems of video production. Moreover, the analog storage of pictorial images on video tape and microfilm is possible and its use will grow. The video monitor included in the workstation can be used directly to view these images because the input waveform is the same for both sources. Optical Character Recognition (OCR) is just beginning to capture and create markets and technology for the input of all sorts of printed matter, pictures and drawings. The data

structers produced by these scanners is compatible with video production. 5a5c

Computer generated video has many advantages. In fact, no technology available today provides the flexibility and effectiveness of video. In the sections below, we shall look in detail into the problem and processes of video. 5a6

2.2 ORGANIZATION OF THE VIDEO WAVEFORM 5b

In order to create the video waveform, the picture is converted into a series of horizontal strips, starting at the top and continuing to the bottom of the screen. These strips, called horizontal scan lines, are made by measuring the whiteness of the picture at each point from left to right. Since the strip is determined by scanning sequentially from left to right, the horizontal displacement (x dimension) is represented by the time from the beginning of the scan line. In the same way, the vertical dimension is determined by the time from the beginning of the first scan line. 5b1

The complete picture could be represented by the horizontal scan lines taken one right after the other; however, it is not. The picture is scanned in two passes called fields. The even numbered scan lines (0, 2, 4, 6, ...) are placed in the first field; and the odd numbered scan lines (1,3,5,7, ...) are placed in the second. Together the first and second field are called a frame. The fields are said to be "interlaced 2:1." 5b2

Since each field must be completed in 1/60 of a second, and two fields comprise a frame, the complete picture is repainted 30 times a second. This is comfortably above the threshold for persistence of vision so that the viewer does not perceive any flicker. 5b3

Each horizontal scan line is separated from the next by a pulse called the horizontal sync. The video signal starts just above 0 volts with black, continues with successively lighter shades of grey, and ends with white at the highest level. Near the horizontal sync pulse, the signal level drops below black (called blacker than black) to a level which insures that no image is produced. The process of suppressing the video signal is called blanking. The actual horizontal pulse is located slightly off center of the horizontal blanking pulse and is negative going. 5b4

The horizontal sync pulse resynchronizes the horizontal oscillator in the tv monitor so that each of the scan lines starts at the same point on the monitor screen. In addition to resynchronizing the horizontal oscillator, the horizontal blanking period allows the beam to retrace back to the left side of the screen before it launches off for the next scan line of the field. 5b5

The horizontal oscillator is used to deflect the CRT beam

horizontally from left to right in a consistent linear way and then snap the beam back to the left during the horizontal blanking period. The vertical oscillator performs the same function for the vertical axis but at a much slower rate. The video waveform provides a sync pulse similar to the horizontal one to resynchronize the vertical oscillator of the tv monitor. Retracing the beam from the bottom right of the screen to the top left takes about 1.25 milliseconds. Horizontal scan lines which would ordinarily fall into this time period are lost. 5b6

Interlace is accomplished by delaying the start of the second field. This phase change is accomplished by continuing the horizontal sync pulses during the vertical sync period. These pulses are called serrations. Near the middle of the vertical period the horizontal sync pulse train is shifted by one half of a horizontal time period. The vertical oscillator has remained constant, but the video information is delayed by one half period. The vertical oscillator causes the beam to move down the face of the CRT by two scan lines for each horizontal period. As a result, the beam of the tv monitor will be lower on the crt by exactly one scan line. For this field the scan lines will be between those of the preceding field. On completion of the second field the delay is removed and the process is repeated. 5b7

2.3 DIGITAL GENERATION OF THE VIDEO WAVEFORM 5c

There are several methods available for the digital generation of the video waveform; however, the two simplest are most applicable to this terminal. 5c1

2.3.1 Bit Map 5c2

As its name implies the screen is represented by a matrix of single bits which can be set on or off to represent lines, areas, and characters. For a full page display this matrix needs to be at least 640 bits wide by 1024 bits high. 5c2a

There are two limitations to this method. 5c2b

1) A large amount of memory is required. 5c2b1

2) The number of CPU cycles needed to create the image in the bit map is considerable. 5c2b2

The first limitation is not too great because the cost of memory is dropping. Even in small quantities the memory cost of a bit map of page size is less than $3000. 5c2c

The second limitation is more troublesome. Since a primary use of the terminal is in the manipulation of text, a second generation method will be used which is more economical in its use of CPU resources. 5c2d

2.3.2 Fixed Cell 5c3

In this method the screen is divided into a matrix of character cells. For full page the organization might be 80 cells wide by 64 cells high. 5c3a

Each cell is built from a number of bits in a matrix. These cells may be of any size and a cell 8 bits wide by 16 bits high is convenient. The screen image is stored in two memories - the refresh memory and the cell or font memory. 5c3b

In order to create the display, a code is fetched from the refresh memory and used as a key to fetch a cell from the font memory. This approach is natural for text work and is employed in most commercial video displays. Character definitions are stored in a fixed "read only" memory; however, if the user needs to define his own special symbols, then a "read/write" memory can be used. 5c3c

3.0 ORGANIZATION OF THE WORKSTATION 6

A prototype workstation is shown in figure 1. Both the tv monitor and the interactive input equipment can be seen. The electronics is packaged so that it can be stored in the worktable pedestal. 6a

Figure 2 shows the organization of the terminal electronics. A microprocessor controls the input gear and the video generators. 6b

The video display has two sections, a character generator and a graphics generator. The character generator displays over 5,000 characters in 64 lines of 80 characters each. This is nearly three times the number available on present "video terminals". Each character cell is defined by a matrix of dots 8 wide by 16 high; 512 of these character cells can be stored in a read/write "font" memory. These definitions can be modified by the terminal processor at any time. The text to be displayed is stored in an 8K refresh memory. (Only 5120 of these characters are visible at one time.) 6c

The graphics generator consists of an array of dots 640 wide by 1024 high. Each bit can be set or cleared by the terminal processor to produce figures of any description. 6d

Both the character generator and the graphics generator run synchronously, so that mixed text and graphics can be displayed. The graphics memory has 96K bytes which can be used to store text when graphics is not being used. This is about 30 typewritten pages. 6e

Preliminary cost estimates show that the device would cost between $4,000 and $6,000 without graphics, or $7,000 to $10,000 with the graphics capacity. The system is designed so that graphics can be added at any time in the field. 6f

Figures 3 and 4 show two pages of text on the display The value of

full page is especially evident in figure 4. In this picture a complete computer subroutine is in view at once. The programmer can see the whole logic at one time without scrolling or paging. 6g

4.0 INTERACTIVE INPUT EQUIPMENT 7

Two special devices are included in the workstation in addition to a keyboard. 7a

On the left in figure 1 is the five-fingered keyset. Combinations of 5 keys are struck to type characters with one hand. On the right is a positioning device called a mouse. Two wheels in the mouse encode the direction and distance as the mouse is rolled on the table with the right hand. This position information is used to position the cursor for both text and graphics manipulation. Three buttons on the mouse are used to indicate command completion and to set the case of the keyset codes. By using the mouse and keyset the user can indicate position and type any character. 7b

The mouse and keyset are not difficult to learn to use. A novice need not use the keyset at first. Together the mouse and keyset are efficient, comfortable, interactive tools for long sessions of editing and manipulation. 7c

5.0 SUMMARY AND CONCLUSION 8

We estimate that the complete terminal with both text and graphics capability can be produced with a selling price of less than $10,000. This low cost is possible through the use of emerging video technology. Moverover, the terminal meets general requirements for a practical computer workstation. 8a

6.0 ACKNOWLEDGMENT 9

This work was supported by the Advanced Research Projects Agency of the Department of Defense and by Rome Air Development Center of the Air Force under contract no. F30602-75-C-0320. 9a

7.0 BIOGRAPHY 10

Robert L. Belleville received the B.S.M.E., M.S., and Ph.D. degrees in 1967, 1969 and 1974, respectively, from Purdue University, Lafayette, Indiana. 10a

His work has ranged from vehical testing at Aberdeen Proving Ground to automatic perception of colors at Purdue. In 1974, he joined the staff of Stanford Research Institute. Dr. Belleville's work at SRI has included interactive graphics, automated publication systems, and interactive workstation design. 10b

Figure 1. Prototype Terminal. 11

JBP 4-Feb-77 15:21 29069
Development input to Chase Proposal
Location: (FJOURNAL, 29069, 1:w)
*****Note: Author Copy*****

<<<<<<<<<<<<<<<<<<<< Text of Cited Document Follows
>>>>>>>>>>>>>>>>>>>>

Absolute Requirements

The system system supports text entry using a Datamedia keyboard which is a standard typewriter keyboard.

The system supports the viewing of text being entered in the form in which it is viewed by the system. Embedded formatting commands known as directives are easily recognizable by the user. They are always surrounded by delimiters that are unique and may be selecteed by the user. For example, .IgD=On; .Ybs; .IgD=Off; is a directive with the delimiters "." and ";".

Required spaces between such logical units as Mr. and Jones can be supported so that lines end at logical breakpoints. a Single character can be inserted between Mr. and Jones so that at the time of printing, it will be translated into a space and the two words will not be on separate lines. words with hyphens with not be broken so that them system ensures that lines end at logical break points.

Upper and lower case characters are supported. All the characters of the ASCII character set are supported.

Errors detected immediately after input are easily corrected. Both single characters and words may be deleted immediately after input by typing either the Back Space word key or the Back Space key. When the Back Space word key is used, the system backs up through the word to the last non-printing character.

The system supports the printing of documents up to 13 inches wide to facilitate the printing of special forms.

Character, word and phrase underlining will be supported.

Horizontal tabulation is supported for indentation and page formatting.

Automatic pagination and page numbering of at least four digits is presently supported.

The user can adjust the spacing of a draft to be either single spaced or double spaced. For the final copy, the user can adjust the spacing to be anything she chooses. Margin settings and tab settings can also be adjusted for the final copy.

By use of directives (Grab, Widow, and Prit) page breaks within sections of text can be prevented.

By the use of the subsystem XTABLE, an operator has the ability to decimally align numeric columns, right justify and left justify columns as desired.

With the use of directives (PES, PBS, PEL, PBL) the user can force as new page at any point.

Directives can be inserted so that page and line layout of a document are the same regardless of the output device used.

There is a hyphenation dictionary that will automatically hyphenate words when desired. This dictionary can be added to by the user.

The system supports the Courier 72 typeface and a pitch of 10 characters per inch.

Predefined formats already exist and can be easily modified to accomodate classes of documents with other specific formats.
The system has the commands, Insert, Delete, Replace, Move and Copy for words, Text, Statements, Branches, and Groups. This provides the ability to add, delete, replace, move and copy words, phrases and sections of text.
Global scans and substitutions can be performed on all or part of a manual to find all occurrences of a character, word or phrase.
Paragraphs or sections are automatically numbered and remnumbered to sixty four levels.
Text (either entire documents or selected segments) that exists in other documents being maintaned by the system can easily be incorporated into the document being worked on. Both the text being worked on and the text being copied can be viewed simultaneously by the user.
Storing and retreiving text can be easily performed by the operator. If an attempt is made to name a new document with a name already being used, the system reports this to the user. Names of documents can be orginated or changed easily at the keyboard.
Space allocation for document storage is handled by the TOPS20 operating system.
All text that has been chaged since a particular prior version can be flagged with any character the user chooses to place in either margin. The user can also have the chaged statements marked with the name of the person that made the change, or with the date or time of the last change. A cumulative count of all the changes can be automatically kept. The user can also chose to print only those parts that had changes made to them.
The user can specify that only parts of a document be printed.
The system has the capaibility of archiving and retreiving documents, deleting documents. The space that is released by these procedures is reusuable.
Searches can be made by the user for both online documents and those that have been stored offline. Those that were transfered to offline storage can be easily reloaded onto the system.
For each online document, the system provides the name that the user has chosen to give it, the owner, the entry date, and online storage location. In addition retention duration can be set by the system administrator, and a particular document can be marked to remain online permanently. Offline storage storage of a document is readily available. All of this information can easily be placed in a report which can be sorted on any one of the items.

Audit and Control Funtional Requirements- Absolute

The system does daily incremntal dumps which will aid in the recovery if a document is erroneously updated or prematurely erased, or if the rcording medium is damaged or destroyed.
The systems allows a user to protect his confidential documents by setting protection to prohibit unauthorized reading, updating or deleting of the text.

Performance Requirements-Absolute

The system today can easily handle the 1981 volume projection of 2000 initial system and 1500 keyboarded pages (26 lines to a page) during a 14 hour day.
The system can easily provide a combination 5 million characters of on-line storage and 100 million characters of archival storage.
The system can simultaneously support at least 12 online terminals.

The mechanism for software restoral exists the the TOPS20 operating system.
The system availability from 7:30 A.M. to Midnight on working days will depend on the CMB operators running the TOPS20 operating system on the DEC machines.

Interface Requirements-Absolute

The system can provide the ability to convert the IBM Mag Card II cards into a format suitable for maintenance on the system.

Financial Control Functional Requirements

It is understood there are no known Financial Control functional requirments.

Special Requirements

The system is presently being used by typists/secretaries.
We presently have a well developed training program, covering all aspects of document entry, editing, storage, retrieval, print scheduling, optional features and efficiency. In addition continual training and additional materials can be supplied.
The system does not provide for continuous data entry even if all other components of the system are down.
By using a Diablo printer, the system has the ability to produce high-quality printed output.
Through the operating system, TOPS20, the system can assign a print priority to output documents and facilitate an express printing facility.
The system will maintain statistics detailing the logon/logoff times, number of keystokes per user between any two points specified by the user.

Desired and Conditional Requirements

The system presently allows an author/operator to define a phrase and assign it an abbreviation which would result in the entire phrase being output wen the document is printed. This can be done through the use of directives or with the Substitute command.
The system can support the following characters: angle brackents, slash, square brackets, braces, slashes, and tilde. It cannot support a bullet, the English pound sign, the plus and minus as one character, the division sign or the fraction one half. The system is based on the ASCii character set.
The system can support documents up to 15 inches wide.
The system can assign a starting page number to any page in a document.
The system can assign qualified page numbers that allow for the insertion of pages into existing documents, maintaining page number correctness.
With the use of directives, it is possible to indicate that a specific amount of blank space be provided at a particular point in a document.
The system can automatically generate a Table of Contents, supplying up-to-date page number references for section and subsection headings.
The system can automatically generate predefined page headings and footings at the beginning of a document, that will continue throughout an entire document. At any point in the document, the user can make a change to the heading that will effect all subsequent headings.
The system can provide for overstrikes as an extra cost option. This is an extension to the underlining capability.
The footnote capability can be provided as an extra cost option.
The index generation capability can be provided as an extra cost option. The author must identify each instance of the words to be

indexed.
The system has the capability to sort various fields in a document according to several sort-keys. As an extra cost option the particular sorting mechanism that CMB desires can be provided.
The system can presently support up-to-date indirect referencing as long as the reference is made to some preceding part of the document.
With the use of the directive IGText, the system provides the ability to maintain optionally printed text within a document. When the document is printed in final copy, this optional text will not be printed.
By using the Diablo printer, the user can insert various type wheels for multiple type faces and sizes. The Diablo also allows for 10 or 12 pitch type.
As an extra cost option the system can support right justification.
The system is presently able to print output on a sequence of preprinted forms without manual intervention.
As an extra cost option the system can support superscripts and subscripts.
The system is presently able to support a "DATE" parameter which supplies the current date on the output document when this paramenter is encountered on input.
Using the command Process Commands, the system executes a predefined series of commands which can include the insertion of some defined text.
The system provides the user with the capability of printing only those paragraphs that have been changed since a certian date, or only those paragraphs that have been changed by a specific author. The user also has the option of printing out only some of the pages in a document.
As an extra cost option the ability to generate trace statistics regarding the use of system components can be provided. This will enable the user to detect if an inordinate amount of the system's resources is being spent on a particular acitivty.
The system presently archives those documents wheich have not be accessed (either for reading or for writing) for a specified period of time, and this time period may be varied by the system administrator.
The system provides the user with the following information about any document: length, date of last update, if the document is in the process of being modified at that time, the person who last updated, and date of original creation.
The system always keeps two versions of a particular document online which allows the user to back off to the prior version if an update was applied by mistake. This back off is possible even if the new version has been printed.
The system presently supports dial terminals which facilitates the sharing of terminals and the use of currently available terminals.
The system allows the user to increase the number of words printed on a page by adjusting the margins and by reducing the space between the printed lines.
The Test Processing capability proposed here is only one system so the problem of multiple systems is not applicable.
The system has a message sending capability which allows a user to correspond with the operator as well as other users. There is also a linking capability which allows users to have a dialogue with the operator and other users. When the user types at a terminal, the person linked to will immediately see the same thing at receiving

terminal. It is also possible to suppress this linking option during printing operations.
The system is not "line numbered" oriented, put each paragraph has a static number that is associated with it throughout the edit operation.

COSTS (one person month costs $10,000.)

Absolute Requirements

** Character, word and phrase underlining must be supported. Underlining can be implemented in the output processor with about one person month effort.

Audit and Control Funtional Requirements- Absolute

Performance Requirements-Absolute

Interface Requirements-Absolute

Financial Control Functional Requirements

Special Requirements

** The system must maintain statistics, by operator, detailing the logon/logoff times, number of keystokes (per session, per document), documents accessed and type of activity (initial entry or revision). A mechanism to count and report the number of keystrokes between any two points indicated by the user can be implemented with one half person month of effort. Logon and logoff time are listed by the system on the operators console, and the time used in a session is displayed to the user at logoff time.

Desired and Conditional Requirements

** The system will have to be modified to allow for overstrikes. This is the same modification that must be done for underlining. The provision of overstrikes will require an additional one half person month of effort beyond the underline implementation.

** The system is presently able to do a limited amount of manipulation with footnotes. With some modification, additional footnote capabilities can be added. Theincorporation of footnote capability into the output processor is a one person month effort.

** The system must automatically generate an author identified keyword index, supplying up-to-date page number references for each keyword. The provision of an indexing capability in the output processor is a one half person month effort.

** The system has the capability to sort various fields in a document according to the sort-key a user specifies. The provision of the particular sorting mechanisms desired by CMB will require one half person month of effort.

** The system must be modified in order to be able to support right justification. The provision of full justification will require one person month of effort.

** with some modification, the system will be able to support superscripts and subscripts. Subscript and super script capability for the diablo terminal only can be provided with one half person month effort beyond the overstrike implementation.

** The ability to generate statistics regarding the use of system components is desired. This will enable the user to detect if an inordinate amount of the system's resources is being spent on a particular acitivty, such as local printing, index generation, command sequences causing multiple passes over the text of a

Basic Opt
1½ + 4½ = 6 mos = $60 K

document.
With about one half person month of effort a trace facility can be provided to indicate portions of the system that could be optimized.

<<<<<<<<<<<<<<<<<<<<<<<<< End of Cited Document
>>>>>>>>>>>>>>>>>>>>>>>>>

Comments: Thanks to POOH.

Costs $10K/person-Mo

	-90%	3084
	-22%	5860
Labor		7150

Direct

Computer	2600
Terminal	100
Phone	50
M&S	100
	2850

Salary	1925
Burden 31%	2521
Ovhd 90%	4791
G&A 22%	
Loaded	5845
Direct	2850
cost	8695
Fee 15%	
price	10,000

NIC [Jake Feinler].PBS;

General

The ARPANET Network Information Center (NIC) is located at the SRI Augmentation Research Center. It is funded under a DCA contract, and staffed by Jake Feinler, Johanna Landsbergen, and Adrian McGinnis. This year, we have added a new staff member, Ken Harrenstien (KLH), formerly at MIT-AI where he was active in designing the mail systems, QMAIL and COMSAT. He will be assisting the NIC with its programming.

The NIC wears several hats with respect to ARC and KWAC. We are Utility customers and use NLS to produce the ARPANET Resource Handbook the ARPANET Directory, and the ARPANET Protocol Handbook. We also use NLS to maintain the Identfile and the NIC/Query data base online. We are members of ARC and contribute to the general research effort, and we provide NIC services to all of the Utility customers who are also ARPANET users (which is most of you).

Memlist

The NIC attempts to make its programming general enough to be of use to KWAC members. We have produced a subsystem called MEMLIST which is capable of producing many kinds of formatted directory listings from the Identfile. It can make listings by group or groups of groups. can merge serveral groups and remove the duplicates, and can sort by several parameters such as group id, individual's last name, organization, mailbox, zip code, etc.

Listings can be output as NLS files, sequential files, COM files (not finished yet), and labels. We still have some work to do on this, but will be glad to show it to you and hope to eventually make it an NLS subsystem.

IDENTFILE

We have done extensive 'clean-up' on the Identfile and encourage all of you to check the entries for individuals at your facilities and let FEEDBACK know of changes.

COMPUTER-PATHWAYS.PBS;

The NIC maintains the Official ARPANET HostName Table, and serves as coordinator for the Network Liaison Group. Recently we put together a table for DCA called [OFFICE-1]<netinfo>Computer-Pathways which attempts to identify host compuers and the various other computers connected to each host. The table also gives the operating system of each computer when known. Hard copies of the table are available or you may copy the online file.

The table is not finalized, and we would appreciate feedback on whether the table is useful and the data correct. (Comments to FEINLER@OFFICE-1)

RESOURCE HANDBOOK

The 1976 Resource Handbook is being printed now. Copies will be sent to the Liaison for each host on the network, and it will also be deposited at NTIS. We regret that we were unable to print a copy for each ARPANET user; however, there are a limited number of extra copies. KWAC members with an interest in receiving a copy should contact FEINLER@OFFICE-1.

The NIC looks forward to working with many of you in the coming year to provide information describing services and resources available on the ARPANET.

NSW Project Status Report

Introduction

This is the last monthly report on SRI's National Software Works contract effort (Contract Number: F30602-75-C-0320). This report briefly summarizes each months events, describes the work and status of each of the items in the contract statement of work (SOW), and lists the reports required.

Summary

July

WM-NLS file interaction design document was produced (26222,).

August

NLS 8.5 was made available for experimental use at ISID.

A Cobol Output system was completed.

The first version of the Frontend in a character at a time, new tool, tenex versions was tested.

September

The version of the Frontend in a character at a time, old tool, tenex versions was tested.

October

The report of the ident system was completed (26669,).

November

The major event of November was our participation in the demonstration of the parts of the National Software Works completed to date, held at Gunter AFS, Montgomery, Alabama, November 17-19, 1975.

The NLS Base subsystem was integrated with the Works Manager File System on 31-Oct-75. The stand alone version of the PDP-11 Frontend was successfully tested 20-Nov-75. The LSI-11 processor arrived 24-Nov-75.

December

This month we participated in a meeting held at COMPASS to discuss the MSG-3 protocol on 15- & 16-Dec-75.

The Programs subsystem integration into NLS 9.0 was completed.

The first proofs of mixed text and line drawings were received from George Lithograph on 12-Dec-75.

January

On 8-Jan-76 we tested the CLI-11 under the ELF operating system on the PDP-11 for the first time (we had previously tested the CLI-11 in a stand alone mode).

The Hyphenation feature in the Output Processor was installed in NLS 8.5 at ISIC.

The set of papers on the Distributed Programming System was completed and made available.

February

Began to use a version of ELF that has the NCP and Telnet in distinct address spaces.

Hosted a visit from the System Design Laboratory (SDL) group of the Naval Electronics Laboratory Center (NELC) to discuss SRI-ARC's NSW work and NLS in general.

Made the revised Graphics subsystem available to NLS 8.5 users.

March

NLS 8.5 was made the standard system at Office-1 and BBNB.

"Frontend Communication Conventions (27719,) were documented.

The MSG-3 interface to the CLI was implemented and is ready to test against BBN MSG-3 implementation. The PCPB8 translation was implemented.

Design of a Telnet-FE and a Input/Output Station (IOS) was begun.

Conversion of user subsystems including the Calculator subsystem was started.

Attended the ADR ELF software demonstration at Gunter AFS 25- & 26-Feb-76.

April

The MSG-3 and PCPB8 interface to the CLI was successfully tested with the Works Manager and MSG-3 implementation on BBNB Tenex. Features for tool controlled switching of grammars were implemented. This permits a tool composed of subsystems (e.g., NLS) to switch the grammar as the user switches between subsystems.

A design report on the Telnet-FE plan was distributed to NSW participants (see--27900,).

A design report on the IOS plan was distributed to NSW participants (see--27899,).

The interface between Graphics and Output to COM was completed. Two features were added to the Graphics system to aid the preparation of viewgraphs. One turns the margin lines off and the other overwrites the drawing at a slight offset to produce darker images on the copy printer.

The Output Processor code was revised for the new L10 compiler. That code was checked out in the NLS 9.0 environment. That is, the Output Processor runs in NLS 9.0, including output to COM.

Additional experimentation with the LSI-11 indicates that the processor is too slow to allow the luxury of high level programming languages. The line processor program was recoded in PAL11X.

ARC hosted a visit by the Systems Design Laboratory (SDL) team from NELC on 21- & 22-Apr-76.

May

The MSG-3 and PCPB8 interface to the CLI was successfully tested with the Foreman/Encapsulator and Old Tools (TECO) and MSG-3 implementation on BBNB Tenex.

The effort to debug the display version of NLS 9 in conjuction with the display features of the CLI was completed for the single window (no graphics) version.

Error reporting standards were adopted (see--28074,).

The stand alone version of the debugger was made available to ARC programmers.

Jim White and Jon Postel attended the Berkeley workshop on Distributed Data Management and Computer Networks arranged by Lawrence Berkeley Laboratory. Jim made a presentation on the Distributed Programming System. Jon made presentations on the CLI/CML user interface and on the Frontend/Backend split environment.

Austin Henderson of BBN visted ARC to discuss the use of the CLI/CML techniques in the design of a mail processing system.

June

The Do All Debugger (DAD) is now in use by ARC programmers with a significant improvement in debugging capability and programmer productivity.

The ARC Documentation group has begun a review of NLS 9.0 to identify inconsistencies between NLS 9.0 and NLS 8.5 and the documentation.

The ELF MSG-3 implementation was begun.

The LSI-11 line process prototype design work has determined that the LSI-11 is not adaquate for the task, we are now investigating the Intel 8080 processor.

A review of the NLS 9.0 code source files is under way to improve the documentation.

A review of the preliminary NSW Exec Command Language was made for consistency with the capabilities of the CLI and CML. The result of the review is (28210,).

A new PROM card was prepared and sent to Gunter for the PDP-11.

July

The NLS subsystems Modify, User Options, and Publish were converted to NLS 9.

The conversion of the Graphics subsystem to NLS 9 was begun.

The "Experimental Graphics User's Guide" (34907,) was published.

The proposed procedure for Frontend-Tool communication via MSG direct connections was documented (28409,).

The integration of NLS and NSW suffered a setback (28474,).

August

The IO station was made available to NELC and Gunter.

A User Profile Users' Guide (28554,) and Data Structure Specification (28553,) were completed.

We are proceeding as directed (see--36513,) to implement a FE-11 BE-10 version of NLS 9, independent of NSW but using the same protocols where possible, with the intent of providing a demonstration of the 10-11 split system at the earliest possible date. We are also proceeding with developments on the PDP-11 to interface with the NSW protocols as they become defined.

September

"Minimum Frontend Specification" (28672,) was published.

Testing of the PDP-11 Frontend with the PDP-10 Backend proceeded to the point that commands specified at PDP-11 user interface were carried out on the PDP-10, and results were displayed to the user at the PDP-11.

The I/O station compatible Receiver Spooler program in Tenex was made available for experimental use.

October

Subset of MSG on the PDP-11 for Frontend-NLS Backend 11 communication.

Our work has been seriously hampered by the reduction of our computer resources.

November

FE-10 to Tool communication using the latest conventions was tested.

Parse functions for filename fillout written.

Video terminal tested with 64 lines by 80 characters in teletype simulation mode.

Our work continues to be hampered by poor computer support. Our work is also delayed due to partially moving our computer support to ISID, a TOPS 20 system.

December

In this final month of the contract we brought many tasks to completion and perpared the remaining documentation called for in the contract. The documentation completed includes the following:

An Introdiction to the Frontend (28743,)
A Guide to the Command Meta Language and the Command Meta Language (28744,)
Frontend System Documentation (27845,)
Frontend program Documentation (28981,)
Debugger Program Documentation (28982,)
NLS Program Documentation (28983,)
The design of a Network Hardcopy Facility (28987,)
Frontend Performance Analysis (29025,)
System Configurations for Interactive Tools (29045,)
Command Sequence Processor Design Specification (29046,)

Statement of work

4.1 The contractor shall install and maintain the VM ELF Operating System on the PDP-11s at AFDSDC and RADC. The software shall include the MSG-3 interprocess communication protocols developed by Computer Associates (COMPASS) under contract F30602-76-C-0094. The AFDSDC software shall also include the software developed by Applied Data Research (ADR) under contract F30602-75-C-0219.

The PDP-11 at AFDSDC has been using ELF operationally since 18-Aug-75. We are devoting additional attention to the ELF operating system to improve its reliability and to understand and incorporate software provided by Applied Data Research. Found and fixed a bug in the ELF system which resulted in a significant improvement in system reliability (21-Jan-76). Brought up a version of ELF with NCP and Telnet in separate address spaces (11-Feb-76). We are waiting for information about the PDP-11 system at RADC. We prepared a new PROM bootstrap program card for the Gunter PDP-11. This was necessary due to changes in the IMP-Host protocol by BBN (20-Jun-76). We are now using a subset of MSG-11 implementation to communicate between the Frontend on the PDP-11 and processes on the PDP-10 (1-Nov-76).

Input/Output Station (IOS): ARC is designing a mechanism to support the use of peripheral devices for the input and output of data to and from the NSW. The IOS will be centered on a PDP-11 ELF system which could simultaneously support a Telnet-FE or a CLI-FE.

A design report on the IOS plan was distributed to NSW participants (see--27899,) 17-Apr-76. The IOS coding began 28-May-76. The IOS

PDP-11 code was in the debugging stage (20-June-76). The IOS is nearly complete (see--28465,) (29-Jul-76). A preliminary version of the IOS was released to the SDL group at NELC on Aug. 4, 1976 and to Gunter AFB on Aug. 20, 1976. The system is in daily use by NELC to transfer tape and card files (2-Sept-76). Work is proceeding on development of a Temporary Tenex Receiver Spooler for this system, and also on implementation documentation (2-Sept-76). Tenex Receiver Spooler is available at Office-1 and SRI-AIC (7-Oct-76). Card reader bugs and network connection bugs were discovered and corrected (1-Nov-76). Debugging continues (1-Jan-77).

4.2 The contractor shall install and maintain the Frontend (FE) software on the ISIC PDP-10X at Information Sciences Institute, Marina del Rey, CA and the PDP-11s at AFDSDC and RADC. The FE software includes the CLI, CML and L10. The CLI shall run on both the PDP-10X and the PDP-11. The CML and L10 compilers shall run on the PDP-10X within the NLS environment.

The Tenex version of the CLI (based on MSG-1) was debugged for new tools (15-Aug-75) and for old tools (1-Sep-75). A PDP-11 version of the CLI (single user, without network communication) has been debugged on the PDP-11 under the ELF operating system (8-Jan-76) (see--27287,). The PDP-10 version of the CLI was changed to use the compacted form of grammars (15-Jan-76), the PDP-10 and PDP-11 versions of the CLI are now compatible. The CML compiler is working at ISIC as is the L10 compiler.

The Frontend Communications Conventions were documented (see--27719,) (10-Mar-76). The CLI was configured to use the MSG-3 protocol for communication and is ready for testing as soon as the MSG-3 implementation becomes available. The PCPB8 translation was implemented. The MSG-3 and PCPB8 interface was successfully tested with the Works Manager and MSG-3 implementation on BBNB Tenex (23-Apr-76). Features for tool controlled switching of grammars were implemented; this permits a tool composed of subsystems (e.g., NLS) to switch the grammar as the user switches between subsystems (30-Apr-76). The CLI was successfully tested with old tools and MSG-3 (28-May-76). Additional parse functions were implemented to provide for the recognition of selection entities (e.g., branch, plex) (21-May-76). Improvements were made to the CML Compiler (18-May-76). Error conventions were adopted for reporting errors arising in the Backend or other NSW components (see--28074,) (28-May-76).

The FE Tool Suppliers Manual is in progress (28-May-76). Began implementation of MSG-3 on the PDP-11 (15-Jun-76). The conversion to use MSG-3 direct connections for old tools (instead of Pseudo Telnet) was started (15-Jun-76). The ARC Documentation group is reviewing NLS

9.0 for consistency with NLS 8.5 and the documentation. This review has pointed out several bugs in the NLS 9.0 grammar and several inconsistencies in the CLI (20-Jun-76). We also reviewed the Preliminary NSW Exec Command Language memo for compatibility and consistency with the CLI and CML. The result of this review is (28210,) (9-Jun-76). The communication mechanism to connect old tools and the Frontend is in debugging, this mechanism is designed to allow the use of a grammar with an old tool (23-Jul-76). A memo was distributed on the scenario for use of MSG direct connection for FE-Tool communication (see--28409,) (21-Jul-76). A small debugger was created to aid in debugging the PDP-11 version of the CLI (see--28450,) (30-Jul-76).

A PDP-11 version of MSG3 is being designed and implemented (2-Sept-76). The MSG3-local process interface has been specified (see--28612,) (2-Sept-76). ICP to the well-known MSG socket has been tested to ISIC, but no MSG-MSG commands have been exchanged (2-Sept-76). Direct network connections have been established to an NLS BE on the PDP-10 using ICP to an NLS Dispatcher (2-Sept-76). The CLI-11 and MSG-11 interface is currently being tested on the PDP-11 using ARC's PDP-11 DDT (2-Sept-76). The CLI-11 has successfully sent and received calls from an NLS BE using raw network connections established by MSG-11 (5-Oct-76). We are currently implementing MSG-MSG commands to set up and synchronize paths and negotiate for MSG direct connections (5-Oct-76).

Continued work on MSG-MSG commands for path synchronization, MSG messages, and direct connection establishment (1-Nov-76). Will soon attempt cross-net debugging with an MSG3-10 (1-Nov-76). Currently debugging the PDP-10 Frontend using MSG-3 telnet connections for old tools. Previously this was done in an ad hoc manner without MSG-3 (1-Nov-76). Bug fixes and improvements have been made on the operating NSW system (1-Nov-76). Implemented latest NSW tool start up and termination conventions on the PDP-10 Frontend (13-Dec-76). Have begun to determine how to implement looping in the Frontend (13-Dec-76).

The following documentation on the Frontend is now being written: the first draft of "An Introduction to the NSW Frontend" has been completed, the first draft of "Guide to the Command Meta Language (CML) and Command Language Interface (CLI)" has been completed, and the first draft of CML and CLI System Documentation is in progress (13-Dec-76).

The new Frontend has been tested with the new Foreman and the new Works Manager; we have successfully run a tool and terminated it (20-Jan-77).

ADDITIONAL WORK --

Telnet-FE: ARC is designing a modification to Telnet that will allow a simple Frontend to communicate directly with old tools as well as the NSW works Manager.

A design report on the Telnet-FE plan was distributed to NSW participants (see--27900,) (17-Apr-76).

We are planning to prepare a specification document for the Frontend that specifies interface between the Frontend, the rest of NSW, and the functional capabilities necessary in a Frontend. The Frontend Specification document is in progress (30-Jul-76). The document, "Minimum Frontend Specification" (28672,), has been written and published (7-Oct-76).

4.3 Using the software items in 4.1 and 4.2 the contractor shall provide the software necessary to support 20 simultaneous users, employing a mix of the following classes of terminal:

4.3.1 Half duplex line-at-a-time via TIP to PDP-10X only,

Has been tested.

4.3.2 Full duplex character-at-a-time,

Works for both new (15-Aug-75) and old (1-Sep-75) tools on Tenex. The PDP-11 version has been debugged in a single user version integrated with the ELF operating system for the PDP-11 (8-Jan-76).

4.3.3 Two-dimensional CRT terminal using the line processor.

Display code for single window (no line drawings) use is now debugged for the Tenex version of the Frontend (28-May-76). Multiple windows now work (20-Jun-76). Work is in progress on line drawing graphics features. Line drawing features complete (7-Oct-76). Code to provide the same capabilities (multiple windows, line drawings, etc.) is being debugged for the PDP-11 Frontend (7-Oct-76). Display module has been debugged to a certain extent, and is being integrated into the PDP-11 CLI and debugged in that configuration (1-Nov-76).

4.3.4 Defered Execution (DEX) cassette tape recorders (cassette program only on the PDP-10X).

The cassette program has been revised. This program is ready to be installed as an NSW tool.

4.4 The contractor shall develop, install and maintain software to the allow the user to easily express executeable command sequences for NSW works Manager (WM) and tools. This 'macro' capability shall include conditional statements and the ability to prompt the user for input.

Some initial design has been done (28-May-76). We reviewed the comments sent by Gil Meyers of the SDL project at NELC on desirable macro capabilities (20-Jun-76). Additional design work has been done--in the process of finalizing the design, but nothing has been implemented as yet (2-Sept-76). Rough draft of User Specifications is being written (1-Nov-76). Beginning work on the Command Sequence System design (1-Nov-76).

4.5 Using the software in items 4.1 thru 4.4 the contractor shall install and maintain the following NLS subsystems as tools within the NSW environment:

4.5.1 Base,

The combination of NLS 9.0 Base, the CLI-10, and communication using the MSG-1 protocol has worked since 1-Sep-75 on a single Tenex. The Base subsystem of NLS was integrated with the Works Manager File System on 31-Oct-75 in spite of an ever changing target. The evolution of the works Manager File System may cause this task to become undone.

This task has become UNDONE (see--27231,)!

Additional work on interfacing to the rest of NSW will be required when MSG-3 or the Foreman becomes available. Work is also necessary to incorporate some form of identification system for statement signatures (see item 4.6).

The display code has been debugged, except for the multi-window and graphics features (28-May-76). Modifications have been made to allow use of the NLS 8 identfile with NLS 9 (28-May-76). Some checking has been done to verify the consistency of user features between NLS 8 and NLS 9 (28-May-76). Conventions were adopted for the reporting of errors from the Backend to the Frontend (28-May-76). The ARC Documentation group began a thorough review of the NLS 9.0 command language and command behavior for consistency with NLS 8.5 and with the NLS documentation. This review also revealed a number of minor bugs (20-Jun-76). Multi-window display functions are now operational (20-Jun-76).

The design of viewspec display implementation was completed and the Frontend work was done (2-Sept-76). Code to communicate with the

frontend via direct message connect was put in, but it cannot be tested until Foreman is there to open the connection (2-Sept-76). MSG-3 protocol interface to the NLS Backend was implemented (2-Sept-76). Continued fixing bugs in rest of NLS 9 Base (5-Oct-76). Parse functions to allow filename fillout have been written (13-Dec-76).

4.5.2 Userprograms,

This task has been initiated. The first userprogram to be worked on is the Modify Subsystem (28-May-76). The Modify subsystem is partially debugged and work has begun on the Publish subsystem (20-Jun-76). Both Modify and Publish have been completed (30-Jul-76). These subsystems have been debugged: Useroptions, Programs, and Calculator (2-Sept-76). These subsystems have been converted but not debugged: Format and AFM Format (2-Sept-76). Calculator, Modify, and Programs have been tested (1-Oct-76). AFMFormat, Format, and Goto Command have been partially debugged (1-Oct-76).

A new design for the Syntax Generator subsystem has been conceptualized and is in the process of being implemented (1-Nov-76). Work on AFMFormat Table of Contents command has been stopped until Sequence problems have been resolved (1-Nov-76). Began using an experimental library subsystem to update NLS subsystems as well as NLS Base (13-Dec-76). (Note: this was done to synchronize changes in the NLS grammar and parse functions with subsystem grammars and parse functions).

The Sendmail system has been brought up and is running (20-Jan-77).

4.5.3 Help,

Work is completed on the conversion of the help subsystem from NLS 8.5 to NLS 9.0 environment. In NSW, the Help subsystem becomes a separate tool. Debugging the display version of Help revealed problems in the sequence generator that make it impossible to run, therefore have begun to rewrite the sequence generator as a co-routine (7-Oct-76).

4.5.4 Graphics,

Based on user reaction to the experimental system, modifications to the command language were made and a few new features were added, including the ability to control the style of typeface for labels and the type of lines (e.g., dashed, dotted, solid) (1-Apr-76). The interface between Graphics and Output to COM is complete (30-Apr-76). Two features were added to the graphics system to aid in the preparation of viewgraphs. One feature turns the margin lines off and the other overwrites the drawing at a slight offset to produce darker images on the copy printer (30-Apr-76). An improvement was made to allow extended lines to include blank or invisible portions (28-May-76). A user's manual is in progress. The user manual "Experimental Graphics User's Guide" (34907,) was published (30-Jul-76). The conversion of the Graphics subsystem to NLS 9 was begun (15-Jul-76).

The coding of graphics display routines was begun (2-Sept-76). A good portion of graphics routines has been debugged (5-Oct-76). Work is being done to make commands more efficient by minimizing traffic between Frontend and Backend (1-Nov-76).

4.5.5 Output Processor,

A hyphenation capability was completed and installed in NLS 8.5 (29-Jan-76), with dictionary manipulation processes and documentation remaining to be done. Users of NLS 8.5 may use the hyphenation feature by including the directive .Hyphenate=On; in an NLS file. The hyphenation procedure depends on a dictionary. This month we have begun to prepare a complete dictionary. The graphics interface to George Lithograph (Singer 6000) is being debugged, the first proofs of mixed text and line drawings were received (12-Dec-75). Improvements in the COM graphics have been made, including better control over line intensity. Improvements in the COM text have been made, including a more accurate computation of character widths which tends to reduce the white space left between words in fully justified text.

The Output Processor code has been revised for the new L10 compiler. That code has been checked out in the NLS 9.0 environment. That is, the Output Processor runs in NLS 9.0, including output to COM (30-Apr-76).

4.5.6 Programs,

The programs subsystem (and the L10 compiler) is intergrated into NLS 9.0 (31-Dec-75).

4.5.7 User Options,

The user options subsystem is split into two parts in NSW. One part functions to set user profile characteristics acted on by the Frontend; this part becomes a separate tool. The other part manages user options associated with NLS, and it will be integrated with the NLS 9.0. Work has been completed on determining which of the existing functions to implement in each part of the new user options tools (1-Nov-75). The Frontend related user profile tool is completed (1-Dec-75). The Backend related user options subsystem is completed (1-Jan-75). Work is now suspended pending the implementation of NSW interprocess communication system (MSG-3) and the features in the CLI. Only a small amount of additional work is necessary. The Backend User Options subsystem is completely converted to NLS 9 (30-Jul-76). The final version of the document "Userprofile Data Structure Specification" (28553,) and the draft of "Userprofile User's Guide" (28554,) were completed (19-Aug-76).

4.5.8 Caculator,

The conversion of the calculator and other user subsystems to NLS 9.0 was started in April. (See 4.5.2)

4.5.9 and Interactive Debugger.

Work on the debugger was started 29-Oct-74. The stand alone version of the interactive Frontend-Backend split multi-process debugger was made available to ARC programmers this month, resulting in substantually improved program testing progress (21-May-76). The Do All Debugger (DAD) is now in use by all ARC programmers working on NLS 9.0. The availability of this debugger has significantly improved productivity (20-Jun-76). Draft of the document "Design and Implementation of DAD, a Multi-Process, Multi-Machine, Multi-Language, Interactive Debugger" was completed (2-Sept-76). Development of DAD continues (2-Sept-76).

Final version of "Design and Implementation of DAD, a Multi-Process, Multi-Machine, Multi-Language, Interactive Debugger" was completed and submitted for consideration to the Hawaiian International Conference on Systems Support (7-Oct-76). Fixed bugs shown by use of DAD (7-Oct-76). Started initial design work to use DAD for debugging the PDP-11 (7-Oct-76).

The Frontend of these tools shall run on the ISIC PDP-10X and PDP-11. The Backend of the tools shall run on the ISIC PDP-10X.

4.6 The contractor shall provide the NLS Identification subsystem data elements to Computer Associates, Wakefield, MA, along with interface specifications necessary to allow the use of the WM Identification subsystem by the Sendmail tool. The contractor shall identify any additional WM features necessary to allow the send mail tool to operate across multiple hosts.

The report was delivered on 13-Oct-75 (see--26669,).

NOTE: No response has been received from Computer Associates. We must have information on the form and use of the NSW identification system for several tasks, for example: sendmail and statement signatures.

4.7 The contractor shall modify the Output Processor subsystems as necessary to provide the following documentation production capabilities:

4.7.1 Preparation of documents containing mixed text and line drawings.

The line drawing program and features have been debugged, but have yet to be integrated into NLS 9.0 and NSW. These features are available to users of NLS 8.5 (30-Oct-75). The Graphics subsystem has been integrated into NLS 9.0, and mixed text and graphics documents can be produced using NLS 9.0 (31-Dec-76).

4.7.2 Page-by-page formatting and proofing of the above on the Tektronix CRT.

The page proofing program and features have been debugged, but have yet to be integrated into NLS 9.0 and NSW. These features are available to users of NLS 8.5. The Proof subsystem has been integrated into NLS 9.0, and mixed text and graphics documents can be proofed using NLS 9.0 and the Tektronix 4014 CRT (31-Dec-76).

4.7.3 Output of the above on magnetic tape in a format compatible with the Singer 6000 and Comp 80 Computer Output to Microfilm (COM) devices.

We have been able to produce text only documents using either of these devices since 1-Aug-75. We have produced mixed text and graphics documents through George Lithograph (Singer 6000) (first on 12-Dec-75), but further debugging is necessary. Production of mixed text and graphics documents via output COM and processed by George Lithograph is now a routine and regular procedure (31-Mar-76). The

output of virtual COM files via the Output Processor in NLS 9.0 has been tested successfully (30-Jul-76). The Graphics and Proof subsystems and the Output Processor have been integrated into NLS 9.0, and mixed text and graphics documents can be produced using NLS 9.0 (31-Dec-76).

4.8 The contractor shall provide on site services at AFDSDC, Gunter AFS AL to assist in applying the NLS tools to AFDSDC documentation and programming problems.

We have a staff member who spends approximately one fourth time at AFDSDC and is otherwise in regular contact with AFDSDC NSW users.

4.9 The contractor shall design and fabricate an advanced development model of a work station to support creation and editing of line drawings and text on commercially available CRTs. The contractor shall use the Digital Equipment Corp. (DEC) LSI-11 family of modules to implement the design.

The phase one mouse and keyset interface is complete. The LSI-11 processor has been repaired after having been out of service for one month. Programs written in L1011 can now be compiled and loaded into the LSI-11. The LSI-11 has now been programmed to act as a line processor for INLS (1-Apr-76). Additional experimentation with the LSI-11 indicates that the processor is too slow and too expensive to allow the luxury of high level programming languages. The line processor program has been recoded in pal11x (30-Apr-76). Our experience to date has indicated that the LSI-11 is too slow, thus we are investigating the Intel 8080 (20-Jun-76).

The prototype 8080 based line processor has been built, basic support software (assembler, cross-net loader, debugger) has been implemented. Work is under way on the line processor program and the current program is about at the state where we discontinued work on the LSI-11 version (30-Jul-76). Work is continuing on the 8080 based line processor (2-Sept-76). The processor is being designed to be integrated into the video terminal (2-Sept-76).

4.10 Reliability -- The contractor shall perform a reliability analysis and prediction of the system per MIL-HDBK-217B, dated September 1974. The results of this analysis shall be submitted to the Government as part of the Interim Report.

NOTE: It is our understanding that this applies only to the advanced development model in item 4.9. (We would appreciate a copy of the specification.)

4.11 All computer programs developed under this effort shall be delivered to the Government.

The software is well under way, and the documentation for delivery to the Government has begun (30-Jan-76). A review is in progress of the NLS 9.0 code files to associate abstracts with procedures (20-Jun-76).

4.12 The contractor shall investigate means of monitoring and improving the performance of the FE. Software shall be provided to aid in error and crash recovery, to measure parameters associated with efficiency of the FE, and to collect statistics on FE use.

A brief report titled "Front End Performance Analysis" (see--29025,) was prepared (21-Jan-77).

4.13 The contractor shall investigate the capability, cost, and response tradeoffs associated with a range of local NSW FE facilities. The study shall include configurations ranging from the current FE to a cluster terminal configuration supported by a local processor(s) with disc and file system. The configurations studied must support video terminals that can display multifont text intermixed with line drawings and an associated raster printer. The contractor shall design and implement the video terminal using the experience and software developed in 4.9. The contractor shall design the raster printer.

Configuration Study

The configuration study has been initiated by investigating the possibility of implementing a "L10 Machine" in microcode. The impact of such an L10 Machine on the range of possible terminal configurations is significant. A comparison of execution times for various machines in the PDP-11 family for the instruction mixes we actually use is in progress (28-May-76).

This study attempts to evaluate the potential for providing NSW tool execution in or near the terminal. Issues such as the computer facility topology and its costs, communication costs, user productivity, system reliability, etc. are being studied (5-Oct-76). This brief study considers several alternative configurations for placing more power closer to the user. Two such configurations have evolved during this study; they will be described and compared in somewhat more detail (5-Oct-76).

Video Terminal

Design alternatives are being explored and costs evaulated. design specification has been started (28-May-76). Some parts have been ordered (20-Jun-76). We are studying the terminal constructed by Forest Basket at SLAC (20-Jun-76). A tentative design is developing--an 8080 based "full page" (72 char x 64 line) display (30-Jul-76). The preliminary design and breadboarding of full-paged 8080 based display is well under way (2-Sept-76). This work continues (5-Oct-76).

Character generation portion of full-page display has been completed and is operational (13-Dec-76). It's capacity is 80 characters per line, 64 lines per screen. All character definitions are stored in writable memory so that a total of 512 different character definitions can be portrayed. Work continues on the implementation of bit map portion of the display (13-Dec-76). Additional work is being done to provide software for management of font sets (13-Dec-76).

Raster Printer

A survey of available equipment is in progress. The font system used with the XGP has been studied (28-May-76).

Deleted Tasks

(4.4) The contractor shall install and maintain version 2 of the Procedure Call Protocol (PCP) based on SRI Journal documents #24459-24462 dated 1-Jan-75. [task modified 16-Sep-75.]

This task was modified to be to document the work completed through July 1975. This documentation was completed 20-Jan-76.

DPS Programmer's Guide (Version 2.5) (26271,)
DPS Implementer's Guide (Version 2.5) (26282,)
DPS Version 2.5 Procedure Directory [SYSGD] (26283,)
DPS-10 Notes (26285,)
DPS-10 Version 2.5 Source Code (26267,)
A High-Level Framework for Network-Based Resource Sharing (27197,) Also distributed as RFC 707.
Elements of a Distributed Programming System (27250,) Also distributed as RFC 708.

(4.5.2) Sendmail

To begin this task we examined our needs and knowledge of the Works Manager and found some gaps. In an attempt to fill those gaps, we prepared a set of questions (see--27128,). The response to those questions was disturbingly unhelpful (see--27231,). Our work on this task remains suspended as agreed in discussions between RWW and DLS at RADC on 19-Jan-76. This task was deleted from the SOW 1-June-76.

Contractually Required Reports

	Due	Done
Project Status Report See -- (26183,)	1-Aug-75	24-Jul-75
Identification System See -- (26669,)	15-Aug-75	13-Oct-75
Project Status Report See -- (26375,)	1-Sep-75	4-Sep-75
Funds Status Report	1-Oct-75	4-Nov-75
Project Status Report See -- (26610,)	1-Oct-75	1-Oct-75
Project Status Report See -- (26807,)	1-Nov-75	30-Oct-75
Project Status Report See-- (27037,)	1-Dec-75	25-Nov-75
DPS 2.5 See -- DPS Programmers Guide [Version 2.5] (26271,) DPS Implementers Guide [Version 2.5] (26282,) DPS Version 2.5 Procedure Directory [SYSGD] (26283,) DPS-10 Notes (26285,) DPS-10 Version 2.5 Source Code (26267,) A High-Level Framework for Network-Based Resource Sharing (27197,) Also distributed as RFC 707. Published in National Computer Conference Proceedings, 45:561-570, AFIPS, June 1976. Elements of a Distributed Programming System (27250,) Also distributed as RFC 708.	1-Jan-76	20-Jan-76
Funds Status Report	1-Jan-76	5-Feb-76
Project Status Report	1-Jan-76	31-Dec-75

See -- (27234,)

Project Status Report	1-Feb-76	30-Jan-76
See -- (27471,)		
Project Status Report	1-Mar-76	27-Feb-76
See -- (27669,)		
Funds Status Report	1-Apr-76	7-May-76
Project Status Report	1-Apr-76	1-Apr-76
See -- (27815,)		
Project Status Report	1-May-76	30-Apr-76
See -- (27993,)		
Project Status Report	1-Jun-76	3-Jun-76
See -- (28190,)		
FE Tool Suppliers	1-Jul-76	3-Feb-77
See -- (28743,)		
Funds Status Report	1-Jul-76	13-Aug-76
Project Status Report	1-Jul-76	10-Jul-76
See -- (28368,)		
Project Status Report	1-Aug-76	12-Aug-76
See -- (28521,)		
Project Status Report	1-Sep-76	7-Sept-76
See -- (28624,)		
Funds Status Report	1-Oct-76	1-Nov-76
Project Status Report	1-Oct-76	8-Oct-76
See -- (28753,)		
Project Status Report	1-Nov-76	2-Nov-76
See -- (28807,)		
Project Status Report	1-Dec-76	14-Dec-76
See -- (28938,)		
Computer Software and Program Documentation including Source Code Listings with Comments and Index of Procedures		
Front End Program Documentation	31-Dec-76	5-Jan-77
See -- (28981,)		
Debugger Program Documentation	31-Dec-76	5-Jan-77

See -- (28982,)

NLS Program Documentation See -- (28983,)	31-Dec-76	5-Jan-77
FE System See -- (28744,) and (28745,)	31-Dec-76	3-Feb-77
LSI-11 Technical Report Video Terminal Development (28989,) High Quality Printer Study (28987,)	31-Dec-76	7-Jan-77
Abstract of New Technology (see--29102,)	31-Dec-76	16-Feb-77
FE Configuration Study (see--29045,)	31-Dec-76	28-Jan-77
FE Performance Analysis (see--29025,)	31-Dec-76	21-Jan-77
Project Status Report	31-Dec-76	16-Mar-77
NLS 9 Cue Card (see--29172,)	31-Dec-76	1-Apr-77
NLS 9 Command Summary (see--29173,)	31-Dec-76	1-Apr-77
NLS 9 User Training Guide (see--29174,)	31-Dec-76	1-Apr-77

Additional NSW Reports (not required)

	Journal	Date
WM-NLS File Interaction Design	26222	30-Jul-75
NSW Protocol Thoughts -- Interhost Communication	26491	17-Sep-75
PCPB8 compaction notes	26502	18-Sep-75
Control characters used in the current CLI	26891	7-Nov-75
FE-BE Split Too Wide	27098	4-Dec-75
THE COMMAND META LANGUAGE SYSTEM	27266	6-Jan-76
NSW Frontend Process Communication Conventions	27272	7-Jan-76
NSW Answers	27373	20-Jan-76
Comparison of ELF and RSX-11 for FE use	27569	10-Feb-76
Miscellaneous Frontend Things	27619	18-Feb-76
ISI-FE: Notes to Wilcznski -- FE problems areas	27635	19-Feb-76
Frontend Communication Conventions	27719	10-Mar-76

Documentation Requirements for Standalone CML/CLI	27739	15-Mar-76
FE deficiencies: answer to LLG's 27098	27743	16-Mar-76
Input Output Station	27899	17-Apr-76
Telnet-FE	27900	17-Apr-76
Minimal FE Thoughts	28046	7-May-76
Questions on Cost Benefit of Some CLI Features	28047	8-May-76
NSW Frontend Process Communication Conventions	28075	13-May-76
CLI features from users' point of view.	28132	24-May-76
CLI Implementation Memo	28184	3-Jun-76
Frontend Meeting at RADC on June 3 1976	28187	3-Jun-76
Review of NSW Exec Command Language	28210	9-Jun-76
FE-Tool Communication Scenario	28409	21-Jun-76
MSG Direct Connections for FE-BE Communication	28374	14-Jul-76
NSW Frontend Packages and procedures	26151	17-Jul-76
Scenario for Tool Initialization and Termination	28409	21-Jul-76
Experimental Graphics User's Manual	34907	30-Jul-76
Stand Alone Frontend Programmer's Guide	28451	3-Aug-76
NSW Tool-Frontend Communication	28462	5-Aug-76
Problems in Interfacing NLS to NSW	28474	6-Aug-76
Record of Communications about Interfacing	28472	6-Aug-76
More on NSW Frontend - Tool Communication	28519	12-Aug-76
Specification for FEOPENCONN and FECLOSECONN	28528	12-Aug-76
Userprofile Data Structure Specification	28553	19-Aug-76
Userprofile User's Guide	28554	19-Aug-76
Minimum Frontend Specification	28672	7-Oct-76
Command Sequence Processor Design Specifications	29046	28-Jan-77

< DJOURNAL, 29192.NLS;1, >, 23-MAR-77 15:10 XXX ;;;; .HJOURNAL="BEV 22-MAR-77 12:19 29192"; Title: .H1="Transcription of Ideas from Brainstorming Session"; Author(s): Beverly Boli/BEV; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: BEV; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES; Origin: < BOLI, IDEAS-FILE.NLS;5, >, 22-MAR-77 12:09 BEV ;;;;####;

.PEL; .PN=PN-1; .GCR;This is an exact transcriptin of the ideas generated in the brainstorming session some weeks ago. In the meeting on March 22 it was agreed that Ken, Jim W. and HArvey would each make a pass at classifying the ideas according to three different schemes. These will be journalized next week, and discussed in two weeks.

Page 1

- Feedback Loop
- User--Feedback--decision maker
- newsletter--responsible person to do list implementation
  - testing
- Feedback loop thru dev.
- Encourage new ideas be sent to feedback
- Compile new ideas and bells & whistle suggestions from Feedback - give to dev. once a week

Page 2

- Compare users ideas vs. developers ideas
- An index to indexes - Kirk
- Digitized voice and video
  - Computer video switching for teleconferencing, etc.
  - Application - Auto parts store
    - Inventory
    - Price sheet
    - Picture of parts
- Data Management System
  - Graphical
  - Tabular
- Video Communication Channels

Page 3

- Keyboard plugged into T.V.
- Personal computing (augmenting)
- Accounting program on chip
- Personal...
  - Accounting
  - Income Tax
  - Bike gear
  - Recipes
  - Phone book
  - People's yellow pages
  - Restaurant rating
  - Buying vs. renting house

Page 4

- Cable TV
  - Town meetings 2-way
- Newspaper online
  - Personalized newsfilters
- Community Computing
  - Voting
  - Shopping
  - Mass transit

NLS Journal
- Filtering of "junk" mail
- Better mail handling

Page 5

Study of productivity at different work stations

workstations
- Workspace
- Lighting
- Paper, pencil & sharpeners
- Telephone & headset
- Audio feedback
- Chairs
- Full text & graphics & color at each station
- Calculator
- Personal keyboard for different finger sizes
- Computer knows phone numbers of others

Asynchronous Processing
- Batch & interactive processing
- Asynchronous reporting to user
- Multiple displays running separate processes
- Multiple processors
- Intelligent agents

Page 6

Formatters
- Desk size display
- Very large high resolution display
- Flat display
- Assign CPU to each window created
- Mixed text & graphics
- Better pointing
- Multi-speed mouse

Interactive process commands
- Decision, looping, etc.
- MACRO command language

User adaptive system
- Analysis
- Change user attitudes toward computer
- User characteristics - behavior - performance

Page 7

Mind manipulation

Benevolent

CAE

Experiment with group using/not using computers

Applications for other groups:
- Lawyers

Job alienation/satisfacton/physical impact of online life

Augmentation & organizatonal change
- Mediated communication
  - roles & org structure

Page 7A

Time & motion studies of workstations

Brainwave sensing

Voice recognition

Microfilm (fiche) interfacing

Multiple output depending on terminals

Formatters

Inplace editors
Rewrite page
Full page - multiple fonts
Forms
OP designed for traditional typographers
Book publishing system
Color
Blue pencil - multiple comment on drafts

Page 7B

Study impact of "reactive" rather than "interactive" computing (i.e., quick response) - also with ideal workstation
Data entry using other medium
rand tablets (sic)
Doodle graphics system
Touch displays
OCR
Portable displays - collapsable
Eye tracking
Bio pointing

Page 8

Software Engineering
Getting users to use tools
Expanded library
Source level debugging & graphical
Requirements Analysis & formalization
Design Analysis & formalization
Language Editors
Standards enforcement
compiler output more "info"
Functonal System guides
Automatic documentation extraction
Programming interface for non-programmers
Better error recovery

Page 9

Software Engineering (continued)
Configuration management
Access to other computers (basic, Fortran, etc.)
Study other interactive & batch S.E. tools (programmer's WB, etc.)

Page 10

Social anthropology (Cf. Journal)
Greater info <-> authority, structure
Interactive stat. analysis & plotting package
NLS (all computer tools) is "hard" - how to make it easier
Reducing comp. mystique
(Is paranoia well founded?)

Page 11

Telework (other experience)
Long distance collaboration
Analyze experience/improve system
(Shared screens, other tools!)
fix it
Energy trade-offs. (Cf, Ray3 in telecommunicating policy)

Page 12

Other application groups
Lawyers
Medical

Legislature
Libraries
Architects
Artists
Engineers
Paralyzed/"handicapped"
Executive branch
national
state
Internatonal groups
UN
OPEC
Cartels
Managers
(Knowledge workers)

Page 13

Transcribing conf. records
Shorthand machine
Management tools:
Decision analysis
Linear Programming
Capital budgeting
Forecasting
NLS - Budgeting
PERT/CPM
Simulation
Baseline/calendar

Page 14

Strategies for areas out of expertise
Joint proposals
Subcontracting
Hiring new people
Integrating tools
Interfacing to existing tools (as opposed to reinvention)
Gardening subsystem keyed to geographical location:
Diagnosis of plant illnesses
What to plant, when
Game hook-ups
Carpools

Page 15

Scientific tools -
Analysis/notation
Rewrite NLS - goals:
1. As efficient as possible. Throw out less cost-effective points.
(eg. rewrite in PL/1 - rewrite in 8080 - code
2. Transportable
3. Provability of code - theorem prove
4. Cheapness
5. Modularity of packages
6. Security

Page 16

Attitudes
Not currently conducive to individual initiative.
How to improve (our ARC) working environment+
Long-term goals:

What are our (Current) areas of expertise/interest?
Where do we go?
What do we do?
How do we filter these ideas?
Record in (ISID) ARC documentaton ideas
(Password ARC)
Use 9

Additions from meeting 3/22
Analyze a "complete" tool system
Augment the window office
Add host name to the origin statement

dd host name to the origin statement

SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL

29 404

FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER
FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER
FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER

$JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404
$JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404
$JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404 $JAKE$29404

Monday, July 4, 1977 16:18:55 Monday, July 4, 1977 16:18:55 Monday, July 4,
Monday, July 4, 1977 16:18:55 Monday, July 4, 1977 16:18:55 Monday, July 4,
Monday, July 4, 1977 16:18:55 Monday, July 4, 1977 16:18:55 Monday, July 4,

< GJOURNAL, 29404.NLS.1, >, 2-Jul-77 14:32 XXX ;;;; Title: Author(s):
Andy Poggio, Don I. Andrews, David S. Maynard, Kenneth E. (Ken) Victor,
Larry L. Garlick, Martin E. Hardy, Rodney A. Bondurant/ANDY DIA DSM KEV
LLG MEH RAB; Distribution: /SRI-ARC([ACTION]) ; Sub-Collections:
SRI-ARC; Clerk: ANDY; Origin: < POGGIO, R-D-GROUP-OUTPUT.NLS.3,
>, 1-Jul-77 18:11 ANDY ;;;;####;

R & D - New Products Working Group New Ideas Survey Results
(RDNPWGNIS)

Here are the results of a survey given to the members of the R & D - New Products Group, hereafter known as the R & D Group. We would like the entire ARC staff to examine this survey and contribute any new ideas that they may have. New ideas should be submitted to ANDY; the deadline is Monday, July 11. We would like to combine these new ideas with those in this survey into a final survey which the entire staff would take. The results of this final survey would be used for short and long range planning, assigning priorities, etc. 1

This survey consists of a lightly filtered union of the ideas presented at the first R & D working Group meeting and those presented at the earlier all-ARC brainstorming session. They have been arbitrarily arranged. 2

The following criteria were applied to the survey ideas: 3

Priority - rated on a scale of 1 to 5, 5 is HIGHEST priority. 3a

cost/resources - rated on a (logarithmic) scale of 1 to 5, 5 requiring the highest cost or most resources. 3b

Time frame - time interval which might be expected to pass before which the new company begins exploring/implementing idea. The limiting factors on time frames might be technological, lack of resources, or others. Note that time frame is defined here as time until start of work. Duration of work should be reflected under cost/resources. The rating scale is 3c

< 6 months: 1 3c1

< 2 years: 2 3c2

< 5 years: 3 3c3

>= 5 years: 4 3c4

A zero rating implies no opinion. 4

Each idea is prefaced by 3 numbers representing the group's mean ratings for priority, cost/resources and time frame ratings, respectively. For example, 5

5.0 1.0 1.0 Continuation of burrito day 5a

means that this idea has been given highest priority, lowest cost and shortest time frame ratings. 6

(survey) 7

R & D - New Products Working Group New Ideas Survey Results (RDNPWGNIS)

prio	cost	time	description	
3.4	1.8	1.4	How to improve (our ARC) working environment	7b
3.2	1.3	1.0	What are our (current) areas of expertise/interest?	7c
4.0	2.4	1.4	Analyze a "complete" tool system	7d
4.5	2.5	1.2	Analyze experience/improve system	7e
1.0	1.5	2.5	Energy trade-offs. (Cf, Ray3 in telecommunicating policy)	7f
3.7	2.2	1.4	Study other interactive & batch Software Engineering tools (programmer's WB, etc.)	7g
2.6	3.2	2.0	User adaptive system	7h
4.0	1.0	1.0	Analysis	7i
4.0	1.0	2.0	Change user attitudes toward computer	7j
4.0	2.0	2.0	User characteristics - behavior - performance	7k
4.0	2.5	2.0	Study impact of "reactive" rather than "interactive" computing (i.e., quick response) - also with ideal workstation	7l
2.2	2.5	2.2	Job alienation/satisfacton/physical impact of online life	7m
2.2	2.5	2.0	Augmentation & organizatonal change	7n
0.0	0.0	0.0	Mediated communication	7o
0.0	0.0	0.0	- roles & org structure	7p
2.7	1.5	2.7	Time & motion studies of workstations	7q
2.2	2.7	2.2	Experiment with group using/not using computers	7r
3.5	2.2	2.2	Study of productivity at different work stations	7s
2.0	3.0	2.4	Touch displays	7t
3.2	3.0	2.0	OCR	7u
1.7	4.5	3.2	Eye tracking	7v

2.0	3.6	3.0	Bio pointing	7w
1.5	2.7	2.5	RAND tablet	7x
1.7	5.0	4.0	Brainwave sensing	7y
2.0	4.2	2.8	Voice recognition	7z
3.2	2.2	1.6	Message Display	7a@
3.8	2.6	1.6	Appointment Display	7aa
2.7	3.0	2.0	Better pointing	7ab
3.2	2.5	2.0	Multi-speed mouse	7ac
4.0	3.0	2.0	Interactive COM	7ad
4.0	3.2	1.6	Better Graphics	7ae
3.2	4.0	2.4	Desk size display	7af
4.0	3.7	2.0	Very large high resolution display	7ag
2.0	3.7	3.0	Flat display	7ah
3.8	3.2	1.6	Mixed text & graphics	7ai
2.8	4.0	2.5	Voice Input/Output/Storage	7aj
2.6	3.7	3.5	Computer video switching for teleconferencing, etc.	7ak
2.8	3.0	2.6	Portable displays - collapsable	7al
2.5	2.0	2.7	Auto Dialup Terminal	7am
4.5	4.2	1.5	Single User NLS Hardware / Mininet	7an
3.3	4.7	2.3	Without External OS	7ao
4.3	3.0	1.3	Local Terminal Power	7ap
4.8	3.2	1.8	Local Statement Editing	7aq
4.5	3.0	1.6	Micro FE	7ar
4.7	2.7	1.7	Mini/Micro FE/Batch NLSBE	7as

4.0	3.7	2.0	NLS On Other Hosts	7at
3.2	2.6	1.8	Multi Host DAD	7au
3.7	2.2	1.7	Distributed Librarian	7av
5.0	4.0	3.0	To Maintain NLS On Multiple Hosts	7aw
3.7	3.2	1.7	Multiple Language Software Workshop	7ax
4.6	3.0	1.0	Integrating NLS And Other Tools	7ay
4.8	2.4	1.2	Software Engineering	7az
5.0	2.0	1.0	Getting users to use tools	7b0
3.0	2.0	1.5	Expanded library	7ba
4.5	3.0	2.0	Source level & graphical debugging	7bb
5.0	3.0	1.0	Requirements Analysis & formalization	7bc
5.0	4.0	1.0	Design Analysis & formalization	7bd
4.0	3.0	1.0	Language Editors	7be
0.0	0.0	0.0	Standards enforcement	7bf
0.0	0.0	0.0	compiler output more "info"	7bg
3.5	2.0	1.5	Functonal System guides	7bh
3.0	2.0	2.0	Automatic documentation extraction	7bi
4.0	4.0	2.0	Programming interface for non-programmers	7bj
3.0	4.0	3.0	Better error recovery	7bk
4.0	2.0	2.0	Configuration management	7bl
0.0	0.0	0.0	Access to other computers (basic, Fortran, etc.)	7bm
4.0	1.7	1.3	workstations	7bn
4.0	1.0	1.0	workspace	7bo
4.0	1.0	1.0	Lighting	7bp

3.5	1.0	1.0	Paper, pencil & sharpeners	7bq
2.5	1.0	1.5	Telephone & headset	7br
1.0	2.0	3.5	Audio feedback	7bs
5.0	1.0	1.0	Chairs	7bt
3.0	4.0	2.5	Full text & graphics & color at each station	7bu
0.0	0.0	0.0	Calculator	7bv
1.0	3.0	4.0	Personal keyboard for different finger sizes	7bw
0.0	0.0	0.0	Computer knows phone numbers of others	7bx
3.0	2.0	2.3	Offline Storage	7by
3.0	2.0	1.7	Replace Journal Books With Cassettes	7bz
5.0	2.3	1.0	Hardcopy Devices For Customers	7c@
2.2	2.2	2.7	Auto Dialup Terminal	7ca
3.0	2.6	2.0	Keyboard plugged into T.V.	7cb
4.8	2.2	1.2	Combine LP And Terminal	7cc
3.6	2.6	1.8	Color Terminal	7cd
4.2	3.5	1.7	Mixed Text And Graphics	7ce
4.4	3.3	1.6	Full Page Terminal	7cf
5.0	2.7	1.0	Speedup Nls9	7cg
3.7	2.0	1.3	Extensible Grammars	7ch
3.8	2.4	1.8	Modularize NLS To Sell Parts	7ci
4.5	3.5	2.0	Rewrite NLS - goals:	7cj
4.3	2.7	1.3	1. As efficient as possible. Throw out less cost-effective points.	7ck
3.0	4.5	1.0	(eg. rewrite in PL/1, DoD1 - rewrite in 8080 - code	7cl
4.3	3.0	2.0	2. Transportable	7cm

1.0	4.0	4.3	3. Provability of code - theorem prove	7cn
2.0	3.0	3.5	4. Cheapness	7co
3.0	3.0	3.0	5. Modularity of packages	7cp
1.5	0.0	4.0	6. Security	7cq
4.2	2.5	1.3	Better Documentation	7cr
3.5	2.2	2.0	Automatic Documentation Generation	7cs
4.2	2.5	1.5	Multiple Levels Of Training	7ct
3.0	1.3	1.3	Keyset Problems	7cu
2.0	2.0	2.0	NLS (all computer tools) is "hard" - how to make it easier	7cv
5.0	1.0	1.0	reduce computer mystique	7cw
2.5	2.5	2.5	Computer Assisted Education	7cx
3.3	3.0	2.5	Teleconferencing	7cy
4.5	1.5	1.0	Augment the window office	7cz
3.0	2.0	2.0	Gardening subsystem keyed to geographical location:	7da
0.0	0.0	0.0	Diagnosis of plant illnesses	7daa
0.0	0.0	0.0	What to plant, when	7dab
2.0	3.0	2.0	Game hook-ups	7dac
0.0	0.0	0.0	Carpools	7dad
4.0	3.0	2.0	Scientific tools -	7dae
4.0	2.0	2.0	Analysis/notation	7daf
0.0	0.0	0.0	Transcribing conf. records	7dag
0.0	0.0	0.0	Shorthand machine	7dah
4.3	3.7	1.7	Management tools:	7dai
3.0	4.0	2.0	Decision analysis	7daj

3.0	3.0	2.0	Linear Programming	7dk
5.0	0.0	1.0	Capital budgeting	7dl
5.0	4.0	2.0	Forecasting	7dm
3.0	0.0	1.0	NLS - Budgeting	7dn
4.0	3.0	2.0	PERT/CPM	7do
4.0	3.0	2.0	Simulation	7dp
5.0	2.0	1.0	Baseline/calendar	7dq
3.5	2.5	2.0	Lawyers	7dr
2.5	4.0	2.5	Medical	7ds
3.7	3.0	2.0	Legislature	7dt
3.0	3.0	2.0	Libraries	7du
3.0	3.0	2.0	Architects	7dv
3.0	3.0	2.0	Artists	7dw
3.5	3.0	2.0	Engineers	7dx
3.0	3.0	2.0	Paralyzed/"handicapped"	7dy
3.5	3.0	2.0	Executive branch	7dz
0.0	0.0	0.0	national	7e@
0.0	0.0	0.0	state	7ea
3.0	3.0	2.0	Internatonal groups	7eb
0.0	0.0	0.0	UN	7ec
0.0	0.0	0.0	OPEC	7ed
0.0	0.0	0.0	Cartels	7ee
4.0	2.5	1.5	Managers	7ef
4.0	2.5	1.5	(Knowledge workers)	7eg

4.0	2.5	2.0	Telework	7en
3.0	3.0	2.0	Long distance collaboration	7ei
3.0	2.3	2.3	Interactive stat. analysis & plotting package	7ej
4.0	3.0	2.0	Surveys	7ek
2.5	4.0	3.0	Community Computing	7el
0.0	0.0	0.0	Voting	7em
0.0	0.0	0.0	Shopping	7en
0.0	0.0	0.0	Mass transit	7eo
2.0	4.0	3.0	Newspaper online	7ep
0.0	0.0	0.0	Personalized newsfilters	7eq
2.0	4.0	3.0	Town meetings 2-way	7er
3.5	2.0	2.0	Personal computing (augmenting)	7es
0.0	0.0	0.0	Accounting	7et
0.0	0.0	0.0	Income Tax	7eu
0.0	0.0	0.0	Bike gear	7ev
0.0	0.0	0.0	Recipes	7ew
0.0	0.0	0.0	Phone book	7ex
0.0	0.0	0.0	People's yellow pages	7ey
0.0	0.0	0.0	Restaurant rating	7ez
0.0	0.0	0.0	Buying vs. renting house	7f@
4.4	1.0	1.2	Add host name to the origin statement	7fa
3.0	3.0	2.0	Doodle graphics system	7fb
0.0	0.0	0.0	Multiple output depending on terminals	7fc
3.5	3.0	2.0	Inplace editors	7fd

0.0	0.0	0.0	Rewrite page	7fe
4.5	3.3	1.5	Full page - multiple fonts	7ff
5.0	2.0	1.5	Forms	7fg
4.0	0.0	1.0	OP designed for traditional typographers	7fh
2.0	4.0	3.0	Book publishing system	7fi
3.3	3.0	2.0	Color	7fj
3.0	0.0	1.0	Blue pencil - multiple comment on drafts	7fk
0.0	0.0	0.0	Microfilm (fiche) interfacing	7fl
4.2	2.3	1.2	Interactive process commands	7fm
0.0	0.0	0.0	Decision, looping, etc.	7fn
0.0	0.0	0.0	MACRO command language	7fo
3.7	3.0	1.7	Asynchronous Processing	7fp
3.0	0.0	0.0	Batch & interactive processing	7fq
3.5	2.0	1.0	Asynchronous reporting to user	7fr
3.0	2.0	2.0	Multiple displays running separate processes	7fs
2.0	0.0	3.0	Multiple processors	7ft
0.0	0.0	0.0	Intelligent agents	7fu
4.7	3.3	2.3	Data Management System	7fv
0.0	0.0	0.0	Graphical	7fw
0.0	0.0	0.0	Tabular	7fx
3.0	2.0	1.0	Strategies for areas out of expertise	7fy
0.0	0.0	0.0	Joint proposals	7fz
0.0	0.0	0.0	Subcontracting	7g@
0.0	0.0	0.0	Hiring new people	7ga

0.0	0.0	0.0	Integrating tools	7gb
5.0	2.0	1.0	Interfacing to existing tools (as opposed to reinvention)	7gc
5.0	2.5	1.0	Journal	7gd
4.5	1.0	1.5	Answer Feature	7ge
0.0	0.0	0.0	Enhancements	7gf
5.0	0.0	1.0	Handle Different File Types	7gg
5.0	2.0	1.0	Immediate Delivery	7gh
4.0	2.0	1.5	Better mail handling	7gi
0.0	0.0	0.0	Video/Audio Information	7gj
0.0	0.0	0.0	Single Write Memory	7gk
3.0	2.0	2.0	Filtering of "junk" mail	7gl
4.5	2.3	1.0	Feedback Loop	7gm
4.5	1.5	1.0	Feedback should go through Development Group	7gn
0.0	0.0	0.0	Encourage new ideas be sent to feedback - give to dev. periodically	7go
0.0	0.0	0.0	Compare users ideas vs. developers ideas	7gp
0.0	0.0	0.0	Filtered/Sorted Feedback Input	7gq
0.0	0.0	0.0	Full Loop	7gr
0.0	0.0	0.0	People Oriented	7gs
0.0	0.0	0.0	Newsletter	7gt

SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL
SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL

FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER F
FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER F
FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER FEINLER F

$JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$4097
$JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$4097
$JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$40974 $JAKE$4097

Monday, July 4, 1977 17:58:08 Monday, July 4, 1977 17:58:08 Monday, July 4, 1
Monday, July 4, 1977 17:58:08 Monday, July 4, 1977 17:58:08 Monday, July 4, 1
Monday, July 4, 1977 17:58:08 Monday, July 4, 1977 17:58:08 Monday, July 4, 1

SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL SRI-KL

Source Files for L10 Support

A list of the source files for L10 compilers and runtime support, as currently maintained at ISIE for NLS9.

Source Files for L10 Support 1

The following files are currently maintained at ISIE in directory <L10>. They constitute the sources for the Tree Meta and L10 compilers, the Tree Meta generated compiler support library, and the L10 runtime support packages. 2

All of these sources pertain to NLS9 -- that is, the compilers compile under NLS9 and NLS9 uses this L10 runtime support package. 3

META9.NLS 4

Sources for Tree Meta compiler. Also contains RUNFIL to load compiler under TOPS-20. Compiles to META9.REL. Runfil is META.RUN. 4a

L10.NLS 5

Sources for the L10 compiler. Branch 1 is the Tree Meta sources, and compiles to L10.REL. Branch 2 and 3 are in L10 language and support the compiler itself, are compiled to LIBE10.REL and LIBE109.REL to be used in XL10 (for NLS8) and L109 (for NLS9) respectively. The runfil is in branch 4 , and produces the L109.EXE compiler. 5a

L10SYMS.NLS 6

This is an L10 source file that contains initialization symbols for the L10 compiler. Output sequential to L10SYMS.TXT. 6a

METOP.TXT 7

A small file containing initialization opcodes for creating the META9.EXE compiler. 7a

OPS.TXT 8

A small file containing initialization opcodes for creating the L109.EXE and XL10 compilers. 8a

LIBE9.NLS 9

The L10 sources for the support library that is used by all Tree Meta generated compilers. It compiles to LIBE9.REL and start loading at location 400010 octal in order to run in the same address space as NLS. An encapsulated or stand-alone version could be loaded at any location ten octal words in from a page boundary, i.e. xxx010 octal. 9a

MINIL10DATA.NLS and MINIL10RUNTIME.NLS 10

Source Files for L10 Support

These contain the MINIMUM code required as runtime support for L10 programs. Basically, they contain procedure call/return, and PCALL support. They are used by the compilers and L10LDR. Compile to MINIL10DATA.REL and MINIL10RUNTIME.REL. 10a

XL10DATA.NLS and XL10RUNTIME.NLS 11

These are the basic runtime support for L10 programs. They contain the procedure call/return, PCALL, signal, catchphrase, multiple general stack, string and pattern matching (FIND) support. Compile to XL10DATA.REL and XL10RUNTIME.REL. 11a

L10LRP.NLS 12

The L10 LIST runtime support package. Requires the STGMGT package. Compiles to L10LRP.REL. 12a

STGMGT.NLS 13

Storage management package. Contains procedures MAKEZONE, GETBLK, FREEBLK, etc. Compiles to STGMGT.REL. 13a

L10LDR.NLS 14

Sources for the L10LDR.EXE loader. Also contains the runfil to load it. Compiles to L10LDR.REL. Runfil is called L10LDR.RUN. 14a

How to use NLS9 with Datamedia and no Lineprocessor

How screen selections are made on a Datamedia (2500) without Lineprocessor. 1

A short description of how to use DNLS9 with a Datamedia 2500 without Lineprocessor and mouse. The technique can be implemented for other brands easily. 1a

Warning: The Datamedias at ARC are modified so that "protected" (bright) characters are "unprotected". If you wish to run on a standard Datamedia, you can but marking and unmarking the screen does not work exactly right. If there is any demand for it, NLS9 can be chaged to use "blinking" for standout on standalone Datamedias, and the problem will go away. 1a1

First of all, NLS9 must know that you have a bare Datamedia. There is no TOPS-20 or TENEX-wide convention for a terminal type Datamedia, so temporarily, terminal type 37 is used. (say TERMINAL 37 to TOPS-20). 1b

As you enter NLS, you will get the message "your terminal baud rate is:". Respond with the baud rate, and expect single character recognition (e.g. type 1 for 1200 buad, and so on). (It may be possible to get the baud rate automatically on TOPS-20, but only for direct connection data lines.) 1c

A screen selection is made by typing a sequence of characters instead of moving the mouse and hitting the OK button. The sequence is <ESC> <position characters> <OK>. While typing this sequence of characters, a "*" appears in the command feedback line to show that you are in position mode. 1d

The position characters are any number of the following: 1e

X and then Y coordinate: 1e1

The attempt is to encode each coordinate as a single character so that only 2 characters specify a point on the screen. The encoding is the same for X and Y. The origin is the character pair 00 and is at the lower left corner. The scale is as follows: The first ten positions are 0 thru 9, the next 26 are a thru z, the next 26 are A thru Z, the next 9 are ! thru) (the shifted digits). 1e1a

The pair 0j is the left edge, top line of the text window. One of the coordinate characters may be a space, in which case that coordinate is unchanged (used for example if you are on the right line or column but wish to move along the other coordinate). 1e1b

The original intention was to tape a scale on the left edge and across the bottom of the screen, to allow the user to make a "good guess" at the coordinates and then correct them with the following scheme. 1e1c

Incremental changes: 1e2

The buttons < > ^ <LF> on the right-hand key cluster are used to position the cursor one position per stroke, in any direction. The intention is that these are only be used for fine positioning of a few character places, since it is rather time consuming to move very far this way. 1e2a

Note: 1e3

At any time in position mode, the user can type coordinate pairs or incremental changes over and over in any order. 1e3a

The selection is completed by typing <OK>, and the screen will be marked as for a mouse selection. There are other ways of getting out of this positioning mode: 1f

CD: You will get out of positioning mode and a CD will be input. The cursor will stay where it is. 1f1

BC: (^A) You will get out of position mode and the cursor position will be restored to what it was when you last entered position mode. 1f2

MASTER CLEAR (^^): Same as system reset on a Lineprocessor. The cursor will stay where it is. 1f3

ESC: Typing two ESC characters will result in one <ESC> being passed as input, plus you will be removed from position mode. 1f4

Please remember the experimental nature of this mode and send your constructive comments to DIA,ANDY,KEV. 1g

Comment: 1h

The Datemedia 4000 presents an opportunity to do this in a much more satisfactory way: The 8080 processor could recognize one key as an escape to position mode, and hence avoid the use of an ASCII character. Or cursor position keys could be used ONLY for positioning. The positioning could be done within the terminal and be instant, and the repeatchar plus cursor buttons may be wonderful to use that way. But I don't know if the cost advantage is very great over the Datamedia/DNLS variety, even after the programming is done. 1h1

How to use NLS9 with Datamedia and no Lineprocessor

Part of this document has been written and edited using a bare Datamedia at 1200 baud. Here are my comments: 1h2

I find myself occationally confused about whether to type in the vertical coordinate first or not. Perhaps the order should be changed? 1h2a

I do not have any margin scale to indicate what coordinate character to type to get the cursor where I want it. They seem to be very important. I end up using the cursor buttons alot because the coordinates are hard to guess. 1h2b