KEV 25-APR-75 11:08 25789

getting meaningful benchmarks

as many of you already know we are in the process of investigating the possibility of getting our computer resources from places other than bbn, in order to be pseudo "scientific" in making a decision about where we will get our resource, we would like to be able to run benchmarks on various computers and then decide which of the cpus does the best for us. it seems reasonable that the benchmarks we use should resemble the typical type of work that we do. therefore, i would like to get many different runfiles from many different people of their "typical" use of nls. 1

to produce these runfiles, use the "Start Record (of session on file) NEWFILELINK Runfil (format)" command. the session(s) that you record should be one that is repeatable. this can be accomplished in one of the following ways (there may be other ways, but these come to mind offhand): 2

have the first command after the "Start Record" command be a "Create File" command (or a "Copy File" command followed by a "Jump Link" to the new file) and do all your editing, etc. with the new file. when you are done with the new file do a "Quit Nls" command (and NOT a "Stop Record (of session)" command). 2a

before giving the "Start Record" command do an "Update File [NEW]". have the next to last command be either a "Delete Modifications" or an "Update File New" command. have the last recorded command be a "Quit Nls" command. if you use this approach, it is necessary to save both the recorded runfil and the base file upon which the edits, etc. were performed. 2b

when you have recorded a seesion, please tell me about it. we are interested primarialy in dnls sessions, but a few tnls sessions would be useful as well. 3

if you have any questions about our intent, or these procedures, please see me, jan kremers, or dick watson. 4

thanx... 5

KEV 25-APR-75 11:08 25789

getting meaningful benchmarks

(J25789) 25-APR-75 11:08;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /SRI-ARC([ACTION]) JHK([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: KEV;

DVN 25-APR-75 20:17 25790

Fifth NSW Documentation Work Breakdown, Plan, and Assumptions

Updates 24395. If you believe any of our assumptiions false,please let me know.

DVN 25-APR-75 20:17 25790

Fifth NSW Documentation Work Breakdown, Plan, and Assumptions

On Wednesday April 23 Beverly, Kirk, Ann, and I met to reallocate and replan NSW documentation work through July 1 on the bases of changes in the plan for what systems will be delivered, and to integrate Beverly into NSW work. 1

We assume the NSW tools of interest to ARC documentation will be: 2

The Frontend-Works Manager
The NLS-9-Editor tool 2a

(From the point of view of the user, the NLS-9-Editor tool will closely resemble NLS-8,5 Base) 2a1

The NLS-8,5 tool 2b

From the viewpoint of the user NLS-8,5 will look very much like NLS-8 except for having certain additional subsystems such as Graphics and Readmail, and some other additions and changes to other subsystems. 2b1

We also assume that NLS-8,5 will include a Help-Query software able to read more than one file. Some doubt exists on this point. We urge Elizabeth, Harvey, and Kirk to dispel the doubt soon. If the help database were inany files on June 15 and had to be integrated into one file it would probably take several person weeks. 3

We prefer a multifile help database and believe it would be easier to convert many files to one rather than vice-versa, but it is more important that we know what the situation will be, than that it be one thing or the other. 3a

On those assumptionse plan the following Help file structure: 4

One file for the Frontend-Works Manager
One file for general information about NLS
One file for each NLS 8,5 subsystem
One file for NLS-9 Editor 4a

We believe we can generate this file easily from the file on the NLS-8,5 Base subsystem in June when we know in more detail what the difference will be. 4a1

The Documentation Work Breakdown, then is as follows: 5

Front End-Works Manager - DVN 5a

Help File: Rough material online 5a1

Introduction: Nothing done 5a2

Scenario: Covered by Primer and Cobol Programming Scenario below. 5a3

Works Manager Cue Card : Not Started. Not so many problems as the NLS-8.5 Cue card because it is much smaller. 5a4

NLS Environment - Kirk/Bev 5b

Help file, largely online from NLS-8 help, will require some editing and revision 5b1

Sequential I/O - Kirk 5c

The command will be added to the Base Subsystem Help file; a scenario can easily be produced from what Kirk has done already.
No Discursive Introduction 5c1

Cassette/Dex - Bev 5d

The Userguide as recently revised by Ann and Jeanne Leavitt will serve for introduction and Scenario. Beverly will reformat it for Help and add it to the NLS Environment file [a low priority task]. 5d1

Sendmail - Kirk 5e

Help file - complete except for minor revisions
Scenario - Nothing done
Introduction - Nothing done
5e1

DPCS - DvN 5f

Help file - Draft begun. It will include without much rewriting the Format, Modify, and Publish subsystem accounts from NLS-8 Help, the information on directives from the output processor users guide, some other concepts. 5f1

Introduction: Based on in the final report. 5f2

Scenario on using format subsystem. Not started. 5f3

NLS-9 Editor/NLS 8.5 Base Subsystem POOH 5g

Help file - Based closely on corresponding part of NLS 8 Help file.
Scenario and Introduction - Primer will serve.
5g1

Calculator Subsystem POOH 5h

Help file-Will be done Friday 4/25 for documentation review.
Scenario - Calculator won't have a scenario
Introduction - Will be done Friday 4/25 for documentation review. 5h1

Programs Subsystem - Kirk 5i

Help file - to be based closely on corresponding parts of the NLS-8 Help file.
Introduction - Not started.
Scenario for COBOL programming - Not started. 5i1

Graphics POOH 5j

Help file - Prtially written.
Scenario - How to make a simple flow chart - Not started.
Introduction - Not started. 5j1

Useroptions Subsystems - Bev 5k

Help file - To be based closely on the corresponding part of the NLS-8 Help file
Scenario - How a programmer would set himself up from Gunter AFB. We need to talk with some one like Larry Crain about this.
Introduction: None 5k1

Readmail: Kirk 5l

Help file - Partially written in the form of Kirk's design document
Scenario and Introduction: to be revised from Kirk's design document 5l1

Letter Program: Kirk 5m

Help file: To be part of the Base Subsystem file
Scenario and Introduction: a draft was accidently destroyed. Kirk will rewrite 5m1

Message Subsystem - Kirk 5n

Help file to be part part of the Base Subsystem File
Scenario and Introduction: Not started 5n1

Interface to NLS Secretarial Functions - Bev 5o

The contract calls for this Item. It will be a collection of documents. Some are covered by documents listed above such as a letter program Scenario. Others include Scenarios for: 5o1

Entering Text Including tabular material
Creating files
Modifying text
Viewing
Formatting
Printing 5o1a

Bev is going to check carefully to integrate this package with the the teaching materials generated by Susan Roetter, Jeanne Beck et all in Washington where appropriate. 5o2

NLS-8.5 Cue Card POOH 5p

This is going to be tough because commands will certainly be changing into June and the lead time on this fancy color print job has been 6 weeks or more. 5p1

NLS-8.5 Command Summary - Pooh 5q

A low priority Item 5q1

NLS-8.5 Glossary. It is not clear if there should be one. If so it is for after July. 5r

We also offered DPCS training and training materials in the proposal. DVN can do the training if it is not covered by what NSW slots receive from Applications. It is not clear if the documentation listed under DPCS and Secretarial interface above will serve. We need to talk to some one like Larry Crain about what they now think they need. 5s

DVN 25-APR-75 20:17 25790

Fifth NSW Documentation Work Breakdown, Plan, and Assumptions

(J25790) 25-APR-75 20:17;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DMB([ACTION] dirt notebook please) RWW([INFO-ONLY] Please Read) LAC([INFO-ONLY] please note 5k and 5s) JHB([INFO-ONLY] please note 5o) JMB([INFO-ONLY] please note 5o) SGR([INFO-ONLY] Please note 5o) DIRT([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT; Clerk: DVN; Origin: < HAMILTON, DVNNSWDOCMTG.NLS;3, >, 25-APR-75 20:04 DVN ;;;;####;

SLJ 25-APR-75 18:54 25791

feedresponse

rita: i just tried (408)255-7950 and it worked perfectly from here....you may want to try ames tip (near here) (415) 964-8990--this is the one i use. let me know if you still can't get through on a good line. sandy 1

feedresponse

(J25791) 25-APR-75 18:54;;;; Title: Author(s): Sandy L. Johnson/SLJ; Distribution: /RH([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: SLJ;

BEV 25-APR-75 19:31 25792

First Review of the Primer and Practice Sending A File Through Sendmail

In reviewing the TNLS Prime the first time, I would suggest some sort of revisions in accordance with the suggestions below* 1

Why not begin the primer with a quick lesson on how to read one's mail? This would have been very helpful to me the first couple of weeks I was learning the system. 1a

My understanding is that Steps 1 and 2, as well as the first paragraph under INSTRUCTION on the first page, all need to be redone for NSW. 1b

Some of the <CTRL>-ch should be listed at the very beginning of the Primer. I am thinking particularly of <CTRL>-a, x, and o (and whatever does delete word). If the new user knew these controls from the beginning, it would save a lot of frustration, and give her a better sense of mastery (i.e., not a victim of the machine that does things she doesn't want it to do). 1c

Insert a command scenario for the second part of step 12 (Insert Statement). 1d

BEV 25-APR-75 19:31 25792

First Review of the Primer and Practice Sending A File Through Sendmail

(J25792) 25-APR-75 19:31;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /DVN([ACTION]) KIRK([INFO-ONLY]) ;
Sub-Collections: SRI-ARC; Clerk: BEV; Origin: < BOLI,
PRIMER,NLS;1, >, 25-APR-75 16:54 BEV ;;;;####;

BEV 25-APR-75 19:41 25793

Previous Mail/Practicing Sending the Same

I sent the item titled "First Review of the Primer etc, in case you'd like to read my suggestions, and so I could practice sending a piece of mail as the Primer suggested. It works. Voila, two pieces of mail in your file. I hope. 1

BEV 25-APR-75 19:41 25793

Previous Mail/Practicing sending the same

(J25793) 25-APR-75 19:41;;;; Title: Author(s): BeVerly Boli/BEV;
Distribution: /DVN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
BEV;

KIRK 25-APR-75 23:56 25794

Alternative tab design

I have postponed journalizing the tab design to applications until this has been resolved.

Alternative tab design

JLE has suggested an interesting alternative way to implement tabs that would provide more flexibility. The way proposed in 25766 would not allow subsequent altering of tab stops to change column position. This is clearly a desireable capability given the difficulty of initially predicting how big you are going to want the columns. If we implemented tabs so that the actual tab character was entered, and then allowed as many BC characters to be entered directly into the file as there were blank spaces created by the tab character (i.e. until you had backed up to the last visible), we could get both manual and automatic right justification. The protrayal generator(s?) would have to be modified to do the right thing whenever BCs occur after TAB. 1

This would also have the added advantage of ensuring that columns would be lined up when using proportionally spaced fonts or fonts of different sizes in COM. A third advantage would be that this implementation would allow upward compatability in text files as well as the code written when statement-dependant tab stops have been implemented. 2

We would lose the ability of seeing when typing exactly what will end up in the file until statement dependant tab stops have been implemented. 3

None of this would matter if we didn't have to have the right justification feature done by July without statement-dependant tab stops. Perhaps it's not worth doing it wrong. These are the issues. Perhaps Crain et al should be the ones to decide. 4

COM compatable
Adjustable columns
Upward compatability after July 5

vs 5a

What you see is what you get 6

vs 6a

All of the above if you can wait until AFTER statement dependant tab stops have been implemented. 7

KIRK 25-APR-75 23:56 25794

Alternative tab design

(J25794) 25-APR-75 23:56;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /EKM([ACTION]) ARC-DEV([INFO-ONLY]) ;
Sub-Collections: SRI-ARC ARC-DEV; Clerk: KIRK;

KIRK 26-APR-75 03:03 25795

Sendmail problems

I have received no Sendmail items since April 15 except for a few that were secondarily distributed to me by MLK April 24 and none since. Also, at least one thing I have sent has not been received by some of the intended receivers. This lack of service has caused me considerable communication problems and loss of work. If you have sent me anything via sendmail, please do not expect me to act on it. Until this problem is completely cleared up, my online address should be via sndmsg to KKELLEY@BBN (dont forget the extra K). Thank you. 1

KIRK 26-APR-75 03:03 25795

Sendmail problems

(J25795) 26-APR-75 03:03;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: KIRK;

JAKE 26-APR-75 19:54 25796

NIC Reports Due

Dear Stoney, 1

This is an attempt as PI of the NIC contract to straighten out the problems associated with past due reports. First let me say that there was some misunderstanding on my part concerning the Quarterly Management Reports. Craig Fields and I discussed this before the proposal was submitted, and it was my understanding that he was willing to forego QMRs for frequent online status reports, and the proposal spelled this out. Due to all the paperwork snarls the NIC contract was only fully approved and 'inhouse' by March of 1975 - nine months after its actual beginning date. All of this time I was under the impression that the statement regarding reports that appeared in the proposal was in effect. 1a

About two weeks ago Spencer Floyd of our contracts office called and asked me if I had submitted a QMR for March. I had to admit that I had not. However, I had called Steve Walker as soon as I knew he was taking over as Project Monitor, and had given him a verbal report of where things stood. Since that time I have had to prepare this year's proposal and meet a rather harrowing schedule with respect to trying to publish a Resource Handbook in a somewhat uncertain environment due to the demise of our PDP-10 and loss of associated peripherals such as the line printer upon which I was quite dependent. 1b

Spencer indicated that QMRs were due beginning with the date the contract was finalized. If that is so, then I owe you one QMR for the last three month period and a final report which I expect will be in your office no later than the end of July and hopefully sooner. I do not at this moment (Sat.) know the status of the CFSR Reports, but I will check this out on Monday and, if the NIC contract is delinquent, I will try to straighten things out. 1c

With respect to a Final Report for 73-74 covering the NIC activities - Mike Kudlick (who was Manager of the NIC during all of fiscal 73-74) provided such a write-up before he left in August of 1974. It is my understanding that this will be included in the final report that Dirk is submitting. Since the NIC was not a separate contract at that time and was not my responsibility, I have not followed this closely. 1d

If I am mistaken about reports that are due (which I well may be) or if there is something else you need with regard to the NIC contract, please let me know and I will do whatever I can to see that you get the missing items. 1e

Regards,
Jake 1f

JAKE 26-APR-75 19:54 25796

NIC Reports Due

(J25796) 26-APR-75 19:54;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /DLS([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) DCE([INFO-ONLY]) DVN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

POOH 26-APR-75 21:38 25797

Calculator help tool and introduction

This is the help tool for the calculator as it stands today. The last branch 'Offline Introduction' should be read offline. Please comment or come talk to me.

Calculator help tool and introduction

Lexicon 1

calc-ident:
##<calculator-file>## 1a

calcfile:
##<calculator-file>## 1b

calc-file:
##<calculator-file>## 1c

file:
##<calculator-file>## 1d

accumulators:
##<accumulator>## 1e

Calculator:
> The Calculator Tool provides a variety of commands that allow you to do simple arithmetic and integrate it into an NLS file. 2

Introduction:
> With the Calculator tool, you can do calculations involving the operations of addition, subtraction, multiplication and division. A special file automatically keeps records of your running total, the numbers you use and the operations you perform. You can have up to ten running totals at one time, and each total is stored in a numbered "accumulator." Any accumulator total can be inserted following any STRING or STRUCTURE you designate in an NLS file, or can replace any STRING or STRUCTURE you designate in an NLS file. You enter numbers by typing them, giving an address (bugging in DNLS) or using a number from another accumulator. You can also express a number in terms of a simple arithmetic expression ie. (3+4-2x4/2) is the same as 10. You can begin performing an arithmetic operation when you are prompted by Num/C: 2a

Numbers recognizable to the Calculator:

123456	-123456	123456-
123,12	$123,00	0,12345-
123,456	($1,123,123)	12,123,123+
0,1	,12-	0,11
+1		

2a1

Performing an arithmetic operation:
> First, give an operator, either the words add, subtract, multiply, or divide, or the signs +,-,X,/. Follow an operator by a signed or unsigned number, a simple arithmetic expression or the address of a number, and an OK. If you omit the

operator, addition is assumed.
The number you gave and its operator are entered into a special file. The designated arithmetic operation is performed on your number and the number in the accumulator you are using. The resulting answer replaces the old value of the accumulator you were using. 2a2

Calculator-File:
> When you first use the Calculator tool, a file named "CALC-IDENT.NLS" is created in your directory. Subsequently, this file is automatically loaded each time you use the Calculator tool. "CALC-IDENT" records the history of your work like the tape on an adding machine. It keeps records of each number you enter, the operations you perform, subtotals, and totals. All items are first level statements. A line of asteriks marks the beginning of each session. Your "CALC-IDENT" file is a standard NLS file and may be printed. It should not be edited, however. The Calculator command "Write" copies this file to another NLS file which may be edited. 2a3

Accumulator:
When you use the calculator, your running totals are stored in "accumulators." The accumulator you are using is shown in your Tty window. When you begin a calculator session, accumulator 1 is automatically loaded (although the number is not shown at this point). Each time you begin a calculator session, each of the ten accumulators is been set to 0.
You may specify a particular accumulator with the command "Use Accumulator" and a number from one to ten. Your Tty window will display the specified accumulator total and its accumulator number.
You may reset to 0 the value of the accumulator you are using with the command "Clear Accumulator". The accumulators you were using when you ended your last calculator session are also available to you. 2a4

Saved Accumulators example:
> Here are five accumulator totals you presently have (Row A) and below that are five accumulator totals (Row B) that existed when you ended your last calculator session.
Row A Acc.1 11, Acc.2 22, Acc.3 33, Acc.4 44, Acc.5 55.
Row B Acc.1 66, Acc.2 77, Acc.3 88, Acc.4 99, Acc.5 95.
If you are using accumulator 3 (value 33), and you give the "Use Saved" command, you will then be using the accumulator with the value 88. If you then give the command "Use Accumulator" and specify number 2, you will be using the

accumulator with the value 77. You will no longer have access to any of the values in the accumulators from row A. 2a4a

Formatting:
You can specify a particular format for your numbers with the Calculator command "Format." You can indicate whether or not you want commas and/or dollar signs. Numbers can have up to 11 digits with with from 0 to 5 places to the right of the decimal. The default format for numbers is 2 places to the right of the decimal and 9 places to the left: 999999999.99. The Calculator Tool has internal checks that prevent you from inputting numbers that do not fit your specified format, or formats that cannot be used for numbers that already exist. 2a5

Commands in the calculator subsystem: 2b

Add CONTENT OK:
> The Calculator command "Add" adds the number you specify for CONTENT to the total of the accumulator you are using. Then, the accumulated total is ready for your next operation. See also: accumulator. 2b1

Clear:
> The Calculator command "Clear" erases the contents of your calculator file or sets the total of the accumulator you are using to zero. See also: calculator file, accumulator, use. 2b2

Accumulator: Clear Accumulator OK:
> The Calculator command "Clear Accumulator" sets the total of the accumulator you are using to zero. See also: accumulator, use. 2b2a

File: Clear File OK:
> The Calculator command "Clear File" erases the contents of your calculator file except for the origin statement. See also: calculator file. 2b2b

Divide CONTENT OK:
> The Calculator command "Divide" divides the value of the accumulator you are using by the number you specify for CONTENT. See also: arithmetic operators, accumulator, using. 2b3

Execute (command in) TOOL
##<nlsum,execute>## 2b4

Evaluate CONTENT OPERATOR OK:
> The Calculator command "Evaluate" performs the operation you speicfy for OPERATOR on the number you specify for CONTENT and the number in the accunulator you are using. The number you

specify for CONTENT may be entered directly from the keyboard or indirectly as a simple arithmetic expression (e.g.: 1+9-2/4) You can skip the OPERATOR by typing CA when prompted by OK/C: Your CONTENT will then be added to the value of the accumulator you are using. If you specify an OPERATOR, the result will replace the value of the accumulator you are using. The number you specify for CONTENT will be followed by an asterisk when entered in your Calculator file. See also: arithmetic operators, accumulator, calculator file. 2b5

Format:
> The Calculator command "Format" allows you to specify the format of numbers for the accumulator register you are using. Each saved accumulator needs a separate format change specification. Your Calculator file will show a format change with the first number after a Format command is given. The default number format is right-justification, 2 digits to right of the decimal and up to 9 to the left, no commas, and no dollar sign. For example: 987654321.00 See also calculator file, accumulator register. 2b6

Commas: Format Commas ANSWER OK:
> The Calculator command "Format Commas Y" places commas in your numbers like this: 999,999,999.99. If you type n for "No", no commas will be inserted. 2b6a

Dollar: Format Dollar (signs) ANSWER OK:
> The Calculator command "Format Dollar Y" places dollar signs ($) in front of each number. If you type n for "No", they will not be inserted. 2b6b

Left: Format Left (justify) OK:
> The Calculator command "Format Left" writes numbers that appear with no spaces preceding them. See also: Calculator Right. 2b6c

Places: Format Places (to the) Right/Left CONTENT OK:
> The Calculator command "Format Places" allows you to specify the number of digits that will be printed to the right and/or the left of the decimal point. The total number of printing digits allowed in a number is 11. Within this limit only 5 can follow the decimal. If you attempt to enter a number containing more digits to the left of the decimal point than the current format specifies, the error message "Format too small for input" is printed, the operation is not performed and you must begin with another command. If you specify a format that is too small for the number in your accumulator, the error message "Format too small for accumulator, FORMAT RESET TO DEFAULT" is printed.

the operation is not performed, and the place format is changed back to the default.
2b6d

Right: Format Right (justify) OK:
> The Calculator command "Format Right" prints numbers that appear with enough spaces preceding them for billions in places. See also: Calculator Left.
2b6e

Goto (subsystem) SUBSYSTEM OK
##<nlsum,goto>##
2b7

Insert (accum following) STRING/STRUCTURE: . . .
> The Calculator command "Insert" writes the value of the accumulator you are usimg into an NLS file. See also: accumulator.
2b8

STRING: Insert (accum following) STRING DESTINATION OK:
> The Calculator command "Insert (accum following) STRING" writes the value of the accumulator you are using following the type of STRING you specify at the DESTINATION you specify. See also: accumulator.
2b8a

STRUCTURE: Insert (accum following) STRUCTURE DESTINATION LEVEL-ADJUST OK:
> The Calculator command "Insert (accum following) STRUCTURE" inserts the value of the accumulator following the type of STRUCTURE you specifs writes DESTINATION you specify. See also: accumulator.
2b8b

Multiply CONTENT OK:
> The Calculator command "Multiply" multipies the value of the accumulator you are using by the number(s) you specify for CONTENT. See also: arithmetic operators, accumulator.
2b9

NUM:
> a prompt that asks for a number. If you follow this by an OK, it is added to the accumulator you are using. Any character that is not a number is simply ignored.
2b10

OKREPEAT <CTRL-B>
##<nlsum, okrepeat>##
2b11

Quit
##<nlsum, quit>##
2b12

Replace STRING/STRUCTURE (at) DESTINATION (by accumulator) OK:
> The Calculator command "Replace" replaces a STRING or STRUCTURE (such as a visible or a branch) with the value of

the accumulator you are using See also: DESTINATION, STRUCTURE, STRING, accumulator. 2b13

Show: 2b14

Accumulator: Show Accumulator (Registers) OK:
> The Calculator command "Show Accumulator" displays for you the values of the ten accumulators you are using. 2b14a

File DNLS only: Show File (in window) DESTINATION OK:
> The Calculator command "Show File" displays for you your calculator file in a window of your screen. While you have this file loaded, you can see the numbers you are entering into the accumulator you are using. 2b14b

Subtract CONTENT OK:
> The Calculator command "Subtract" subtracts the value you specify for CONTENT from the value of the accumulator you are using.The Calculator subsystem command "Subtract" subtracts the value See also: arithmetic operators, accumulator. 2b15

Total OK:
> The Calculator command "Total" copies the value of the accumulator you are using to the end of your calculator file. The formatted value will be typed. See also: calculator file, accumulator. 2b16

Use:
> See also: How. 2b17

Accumulator: Use Accumulator (number) CONTENT OK:
> The Calculator command "Use Accumulator" allows you to specify for CONTENT the number of the accumulator you wish to use. You can specify any of the ten accumulators that are in the register of accumulators you are using. See also: accumulator, Use Saved (Accumulators). 2b17a

Saved: Use Saved (Accumulators) OK:
>The Calculator command "Use Saved" allows you to use the accumulator register you were using when you finished your previous session with the calculator. After giving this command, you will be using the accumulator from your last session that has the same number as the one you are presently using. See also: accumulator. 2b17b

Write (new) File CONTENT OK:
> The Calculator command "Write (new) File" creates a new file named whatever FILEADDRESS you specify for CONTENT. It will be an exact copy of your Calculator file. The Calculator depends

on the information, structure, and format in the Calculator file. It is not possible to use the Calculator if this structure or format has been changed in any way. A new file may be edited in any way you desire. "Write File" also clears the calculator file of all entries. Therefore any subsequent "Write File" makes a new file containing only those entries since the most previous "Write File". See also: Calculator File, CONTENT, FILEADDRESS, file. 2b18

X: x command
##<multiply>## 2b19

star command: *
##<multiply>## 2b20

plus command: +
##<add>## 2b21

minus command: -
##<subtract>## 2b22

slash command: /
##<divide>## 2b23

semicolon command: ;
##<semicolon>## 2b24

% 2b25

Offline Introduction 3

The commands in the Calculator subsystem allow you to use addition, subtraction multiplication and division and to integrate your totals into NLS files. This introduction describes some general features of the Calculator. For a complete description of the commands and concepts, go to the subsystem Calculator and type "H" for Help. 3a

The Calculator subsystem can be useful in a variety of ways. It can help in planning and preparing budget reports, filling out forms that involve numbers such as an income tax form, and doing simple arithmetic procedures where the numbers keep changing. 3b

Below are listed four categoires of things you can do with the Calculator. Below each category are listed the commands that correspond to that category. At the end of this introduction, you will find an alphabetical list of all the commands that are unique to the Calculator subsystem. You still have available the universal commands although they are not listed. 3c

Calculator help tool and introduction

A. Perform arithmetic operations:
Add, Divide, Evaluate, Multiply, Subtract, Total, 3c1

B. Format numbers with dollar signs, commas, and change the location of the decimal point:
Format Commas, Format Dollar, Format Left, Format Right, Format Places 3c2

C. Make use of other totals from your present Calculator session session or the most previous Calculator session, and erase totals you have stored:
Use Accumulator, Use Saved, Clear Accumulator, Clear File, 3c3

D. Write a total in another NLS file or edit the file that contains the record of your calculations:
Insert Accumulator, Write File 3c4

Once you go to the Claculator subsystem, the prompt characters NUM/C: let you know that the calculator is waiting for a command. NUM: indicates you can type in a number while C: wants you to give a command word. 3c5

A special file automatically keeps records of your running total, the numbers you use and the operations you perform. You can have up to ten running totals at one time, and each total is stored in a numbered "accumulator." 3c6

The Calculator subsystem is easy to use for simple arithmetic calculations. Here is a one way to do an arithmetic problem. 3c7

1. At the herald, give an operator, either one of the command words Add, Subtract, Multiply, or Divide or one of the signs +,-,X,or/. (If you omit an operator, addition is assumed.) 3c7a

2. Follow the operator by a number. (See Number below.) 3c7b

3. Follow the number by OK. 3c7c

4. The designated operation is performed on your number and the total in the accumulator you are using. The resulting answer becomes your new total and you are ready to begin your next command. (See Using Totals below.) 3c7d

Numbers: They can be entered in the following ways:
Typing them in;
Giving an address of a number (bugging in DNLS);
specifying an accumulator where a number is stored;
Expressing a number in terms of a simple arithmetic

expression
Example--(3+4-2x4/2) is the same as 10. 3c8

Using Totals: The total you have in an accumulator can be inserted following any STRING or STRUCTURE you designate in an NLS file. It can also replace any STRING or STRUCture you designate in an NLS file. 3c9

Format: You can also specify a format for your numbers. You can indicate whether or not you want commas and/or dollar signs and where you want your decimal point to be. The Calculator subsystem has internal checks that prevent you from inputting numbers that do not fit your format or formats that cannot be used for numbers that already exist. 3c10

Alphabetical List of the commands that are found only in the Calculator subsystem: (You still have available any of the universal commands.) 3c11

Add CONTENT OK 3c11a

Clear Accumulator OK 3c11b

Clear File OK 3c11c

Divide CONTENT OK 3c11d

Evaluate CONTENT OPERATOR OK 3c11e

Format Commas ANSWER OK 3c11f

Format Dollar (signs) ANSWER OK 3c11g

Format Left (justify) OK 3c11h

Format Places (to the) Right/Left CONTENT OK 3c11i

Format Right (justify) OK 3c11j

Insert (accum following) STRING DESTINATION OK 3c11k

Insert (accum following) STRUCTURE DESTINATION LEVEL-ADJUST OK 3c11l

Multiply CONTENT OK 3c11m

Replace STRING/STRUCTURE (at) DESTINATION (by accumulator) OK 3c11n

Calculator help tool and introduction

Show Accumulator (Registers) OK 3c11o

Show File (in window) DESTINATION OK 3c11p

Subtract CONTENT OK 3c11q

Total OK 3c11r

Use Accumulator (number) CONTENT OK 3c11s

Use Saved (Accumulators) OK 3c11t

Write (new) File CONTENT OK 3c11u

You may also use any of the following arithmetic signs as dcommands 3c11v

x command: X 3c11v1

star command: * 3c11v2

plus command: + 3c11v3

minus command: - 3c11v4

slash command: / 3c11v5

Calculator help tool and introduction

(J25797) 26-APR-75 21:38;;; Title: Author(s): Ann Weinberg/POOH;
Distribution: /BEV([ACTION]) KIRK([ACTION]) DVN([ACTION]) ;
Sub-Collections: SRI-ARC; Clerk: POOH; Origin: < HELP,
CALCULATOR,NLS;7, >, 26-APR-75 21:33 POOH ;;;;####;

DVN 27-APR-75 23:45 25798

Response to Suggestions about NSW Primer

This item responds to (journal,25792,)
Cane Pooh, Bev, Kirk and I meet on Tuesday afternoon to discuss applicants and set up our internal review procedures?

DVN 27-APR-75 23:45 25798

Response to Suggestions about NSW Primer

I got your message, I think the scenario of the readmail system is listed as a seperate item unter the readmil system documentation, I agree about control characters and a scenario for step 12 1

DVN 27-APR-75 23:45 25798

Response to suggestions about NSW Primer

(J25798) 27-APR-75 23:45;;;; Title: Author(s): Dirk H. Van Nounuys/DVN; Distribution: /BEV([ACTION]) DMB([ACTION] dirt notebook please) POOH([INFO-ONLY]) KIRK([INFO-ONLY]) &DIRT([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT; Clerk: DVN;

Online-offline documentation procedures

The best way to create a file that looks good on paper and still works in the help system (with minimum effort of translation between the two uses) is to create the file in help online format with output processor directives such that when output processed, it looks good on paper and when the directives are deleted, it works in the help system. All corrections and updates are made to these source files which contain output processor directives. Print files and help files are generated from these sources. Thus the maintenance of a single source provides both types of documentation via our augmentation tools. 1

I think the best online filing system for maintenance and user retrieval is to store each different type of file in a different directory. The files in help online format with directives deleted would be in directory help. The output processed sequential text files should be in directory userguides (or xuserguides in the case of documentation for experimental versions). The source files should be in a third directory not advertised to users. I propose creating one with the name "xhelp". At some point when the experimental version becomes the running system and we want to work on the next experimental version, we will want to create the directory "xhelp" anyway. There is a good chance that by that time directives can be invisibly imbedded in the text of the same file that is accesed by the help command. 2

Online-offline documentation procedures

(J25799) 28-APR-75 04:01;;; Title: Author(s): Kirk E. Kelley/KIRK; Distribution: /DVN([INFO-ONLY]) POOH([INFO-ONLY]) BEV([INFO-ONLY]) DMB([INFO-ONLY]) ; Sub-Collections: SRI-ARC;; Clerk: KIRK;

BEv 28-APR-75 14:46 25801

Social Calendar

Ann--I just sent you a rather strange message that started out as this one does, but the text reads "essage." Guess how that happened. Anyway, here's a second attempt at using the marvels of electronic communication to talk to someone about 20 ft. away. Is there a day this week when we could get together for lunch? Let me know. --Bev 1

BEV 28-APR-75 14:46 25801

Social Calendar

(J25801) 28-APR-75 14:46;;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /POOH([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
BEV;

JBP 28-APR-75 16:38 25802

MSG a subsystem

The TENEX program MSG for reading and processing mail is now a subsystem. --jon. 1

JBP 28-APR-75 16:38 25802

MSG a subsystem

(J25802) 28-APR-75 16:38;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

BEV 28-APR-75 17:12 25803

Weekly Report, Week Ending 4/25

Dee--Would you please put the Weekly Report in the DIRT Notebook?
Thanks,

Weekly Report, Week Ending 4/25

The week ending 4/25/75 1

POOH 1a

made revisions on the cue card and sent that off to be printed 1a1

worked on the calculator help file and introduction 1a2

interviewed several perspective candidates 1a3

had several meetings to discuss status of NSW and work allocation 1a4

KIRK 1b

finished the design/help-documentation for the new tabs implementation 1b1

finished the revision of readmail for review by applications 1b2

re-wrote and re-did the glossary formatting program 1b3

answered questions, discussed issues 1b4

archived the previous weekly reports 1b5

DVN 1c

The final report returned from SRI editing. They suggested substantial rewriting to make it flow more smoothly. Dick rejected that suggestion. During next week we will read in their editing suggestions in detail, creat a glossary, and get the references into shape. At that point it can go as a draft to Rome 1c1

The NLS-8 Glossary went to SRI editing for copy proof. It should be back early next week and we should be able to send a complete draft to COM at the end of next week or early the following week. 1c2

Inter viewd several applicants 1c3

With KIRK, POOH, and BEV replanned the NSW documentation effort through July 1, for the last time I hope. The plan appears as (,25790) 1c4

1c5

Bev 1d

BEV 28-APR-75 17:12 25803

Weekly Report, Week Ending 4/25

Completed changes in the NLS-8 Glossary. 1d1

Met with Dirk, Kirk, and Ann in two meetings to plan NSW documentation through June. 1d2

Began reviewing TNLSPrimer and other user documentation (mostly scenarios)put together by Susan et al. Worked through materials on TNLS to see how their usefulness strikes me as a naive user. 1d3

Note: henceforth, lets create each new report at the begining of this file. 2

Old reports archived in <arcdocumentation, winter-74,wklyrpt,> 3

Weekly Report, Week Ending 4/25

(J25803) 28-APR-75 17:12;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /DMB([ACTION]) DVN([INFO-ONLY]) KIRK([INFO-ONLY
]) POOH([INFO-ONLY]) DIRT([INFO-ONLY]) ; Sub-Collections:
SRI-ARC DIRT; Clerk: BEV; Origin: < ARCDOCUMENTATION,
WKLYREP.NLS;56, >, 27-APR-75 23:58 DVN ;;;;####;

DPCS Meeting

coelenterata are transparently the sole of jelly fish

DVN 28-APR-75 22:52 25804

DPCS Meeting

I'm game for the DPCS meeting any time Friday or next week, I would rather see it sooner than later. I undrstood participants would include some of the follwoing: Norm Nielsen, Pat Whiting-O'keefe, Tom Humphry, Robert Lieberman, Bob Bellville, and Elizabeth Michael. 1

DVN 28-APR-75 22:52 25804

DPCS Meeting

(J25804) 28-APR-75 22:52;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JML([ACTION] books look neat, thnaks) ; Sub-Collections: SRI-ARC; Clerk: DVN;

DVN 28-APR-75 23:20 25805

NIC Reporting for '72-'74

Duane, 1

DVN 28-APR-75 23:20 25805

NIC Reporting for '72-'74

(J25805) 28-APR-75 23:20;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DLS([INFO-ONLY]) JAKE([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) DCE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DVN;

JBP 29-APR-75 01:52 25806

'MSG' mail reading and processing program documentation

The MSG mail program is installed as a TENEX subsystem at BBNB.

MSG DOCUMENTATION 1

MSG is a program for reading, writing, and subsectioning files which have a message file format. It is very simple and straightforward to use. Commands are initiated by typing one character, which causes the program to type out the rest of the command name and wait for input from you. 2

Before the commands are described, there are a few general statements about how MSG works and some conventions used in describing the commands that you should know about. The prompt characters letting you know that MSG is waiting for a command character to be typed are "<-". When MSG is started up (by typing MSG<return> to the EXEC) it will first try to read your MESSAGE.TXT file in your directory. If it does not exist it will say so and wait for a command to be typed. If you have a MESSAGE.TXT, it will scan it and type out the header information (i.e. the date, from, and subject fields) for each message since the file was last read, preceded by a MSG sequentially assigned message number. These message numbers are used in association with the various commands. 3

However, if you started MSG by typing MSG<space> to the EXEC, it will ask you for a file to be read. Typing an escape as the first character will cause MESSAGE.TXT to be typed out, and confirmation requested from the user to ensure that that was what was intended. Once a file name has been specified and positively acknowledged, then the same information as described in the previous paragraph will be output to your terminal. 4

When reading a message file in MSG, either when starting up MSG or with the Read command described below, the file must be in the so-called message file format. If MSG recognizes that the file does NOT conform to this format, you will be told so. However, you will be given the opportunity to keep everything that has been read so far, but NOT overwrite the 'bad' file. These two exceptional circumstances and some suggestions for getting around them are described at the end of this manual. 5

The following conventions and symbols are used in the command descriptions below. There are only six types of input MSG expects: 6

(1) a MSG command character
(2) a message number sequence
(3) a TENEX file name
(4) a confirmation character
(5) a message destination list (like SNDMSG)
(6) a message body. 6a

To abort output to the terminal type ^O (control-O). If MSG does not

understand your input, it will return to command input mode, or reprompt you. The following are symbols and their associated meanings used in the command descriptions: 7

<FILE-NAME> 8

Stands for any TENEX file descriptor, including TTY: or LPT:. If you are requested to input a file name, the appropriate TENEX confirm will be given (e.g. [old version]). 8a

<MSG-SEQUENCE> 9

This input is promted by the string (message sequence) in verbose typeout mode. A sequence of message numbers has the following format. 9a

(1) Any single message number. 9a1

(2) Any two numbers separated by ">" or ":". This means message numbers delimited by the two outside numbers (e.g. 2>5 means messages 2,3,4,and 5 in that order). NOTE: if the first number is greater than the second number, it means the sequence in reverse order (e.g. 5>2 means messages 5,4,3, and 2). 9a2

(3) A pair of numbers seperated by "-". This is so that the standard interpretation of the string "21-4" means message numbers 21, 22, 23, and 24, and "24-1" is an error. 9a3

(4) Any sequence of the previous three types separated by commas. This is the way to group several non-adjacent messages together. For example: 1,3,5:7,10 means messages 1 and 3 and 5 through 7 and 10. 9a4

<MSG-SEQUENCE> of the types described above are ALWAYS terminated by <return>. 9a5

(5) However, there are special types of message sequences. All are determined by the first character that you type in the <MSG-SEQUENCE> stream. The following are the ten possibilities: 9a6

a. <escape> is typed which causes the current message number to be echoed to you and the relevant process performed on only that message.

b. <control-I> is typed which causes the previous completely specified <MSG-SEQUENCE> to be echoed and processing done on that message stream.

c. R which stands for "Recent messages" only.

d. O standing for "Old messages" only.

e. A standing for "All messages" and which is equivalent to 1:(last message number).

f. D standing for "Deleted messages". This is valid ONLY in the context of the Headers, Undelete, and Delete commands. Everywhere else, the headers of the deleted messages will be printed. Of course, you can delete the typeout of those headers by typing control-O.

g. U standing for "Undeleted messages".

h. I standing for messages in inverse order (you will be asked if you have both recent and old messages if you want the old messages in addition to the recent ones).

i. S for "Subject field search for string" which asks you to provide a string which will be used as a mask match on the subject field of the message headers.

j. F for "From field search for string" which is like S but searches the Author field of the message headers instead.

Types (i) and (j) require you to type a string terminated by <return>. Typing just a <return> (i.e. the null string) means that searching is not to be performed. Otherwise, the search will be performed on the string typed up to (not including) the <return>. The string you type must be an exact match to some substring of the appropriate field, but all alphabetic characters are treated as being upper case. (Note: carriage-retuns in the subject field of the header listing are ignored.)

In the command format below, everything that the program types will be lower case and everything you type will be in UPPERCASE. This is not the case when using MSG, but is used here for clarity. 10

MSG COMMANDS 11

Commands to Manipulate Message Headers 11a

<= Headers (message sequence) <MSG-SEQUENCE> 11a1

The headers for messages will get typed out as per those defined by the message sequence typed. Headers typed which are deleted have an asterisk printed before the header for

that particular message, In order to get the length of the message typed out along with the header, use the I command.

<= Delete (message sequence) <MSG-SEQUENCE> 11a2

This command will delete from the header the information pertaining to the messages specified by <MSG-SEQUENCE> (although it will not reuse the message numbers). NOTE: This command does nothing to the actual message file, but only deletes it from the list of known messages. This command does however effect message numbers specified in later commands in the following way. If you have deleted message number 5 and then try to "Type" or "Put" message number 5 either directly or implied by the use of the ":" option, that message will NOT be included.

<= Undelete (message sequence) <MSG-SEQUENCE> 11a3

Of course! If you can delete a message, you certainly ought to be able to undelete it. This command restores to the header information concerning the messages specified by <MSG-SEQUENCE>.

Commands to See and Move Messages 11b

<= Type (message sequence) <MSG-SEQUENCE> 11b1

This command will type on your terminal the messages specified by <MSG-SEQUENCE>.

<= Put (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME> 11b2

This command will put the messages specified by <MSG-SEQUENCE> into the file specified by <FILE-NAME>. If the file does not exist, it will create that file and write the messages into it. If the file already exists, it will append the messages to the messages already in the file. This command is useful if you want to keep separate files containing messages concerning different topics.

<= Move (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME> 11b3

This command is a convenient combination of the Put and Delete commands. It will first put the messages into the file specified and then delete them from the header information.

<- List (message sequence) <MSG-SEQUENCE>
on file: <FILE-NAME> 11b4

Lists all the specified messages on the file specified. If you are listing more than one message, there is a preface page with the headers for those messages, and you will be asked if you want each message on a separate page. The intension of this command is to allow a user to obtain a reasonable hard copy listing of some messages. (Note: the preface page of headers will have the length of each message included depending on the setting by the I(nclusion of length in header) command.)

Commands to Update Your Message Files 11c

<- Overwrite old file <FILE-NAME> [confirm] 11c1

This command will overwrite the current file (specified by <FILE-NAME>) reflecting the fact that you have deleted messages. That is, if you delete message 2 and then "overwrite" your file, message 2 will disappear from that file. It also rereads your file renumbering your messages.

<- Quit [confirm] 11c2

This command returns you to the TENEX exec without rewriting any file. (Almost equivalent to typing control-C).

<- Exit and update old file <FILE-NAME> [confirm] 11c3

This command is another way to overwrite your old message file, but instead of rereading the file it returns you to the TENEX EXEC. This is equivalent to doing a overwrite followed by a Quit, but without the overhead of rereading the file.

<- Write file <FILE-NAME> sorted by message arrival time 11c4

This is similar in nature to the Overwrite command, except that the messages are sorted into ascending sequence by their arrival time before the overwriting is attempted. The file is then rescanned.

Commands to Read Other Message Files 11d

<- Read file name: <FILE-NAME> 11d1

You can use MSG on any file which has a message format.

This means you can peruse or modify files created with the "put" command. If, for example, you have a file containing messages pertaining to MSG problems, you can read it to make sure you've taken care of them. Read is the command which lets you read files other than MESSAGE.TXT. It also prints out the recent header information for that file.

Commands to Sequence through the Messages 11e

<= Current message is nn of mm messages.
in file: <FILE-NAME> 11e1

This command tells you (1) the number of the current message, (2) the total number of messages, and (3) the file name of the currently active file. The current message is either the last message typed on your terminal or, if you have not typed one yet, either 1 if you had no recent messages or the oldest recent message if you had recent messages. This command will let you know where the Next and Backing up commands will start. Finally, it will tell you what the currently active message file is.

<= Go to message number: <number> 11e2

This will allow you to change the Current message number explicitly. If <number> is not in the range of acceptable numbers (i.e. it is < 1 or > than the number of messages in the file), or you did not type a number, you will be told and the Current message number will not be changed.

<= Next message is: 11e3

This command types the next message (Current message number + 1) if it is not a deleted message or you are at the end of the list of messages. The Current message is always incremented.

<= <line feed> 11e4

Same as Next. Types the message following the current message, and sets the Current message to be that message.

<= Backing up -- previous message is: 11e5

This command always types the previous message (i.e., Current message number - 1). It is the inverse of the Next command. It always decrements Current message number.

<= " 11e6

This is equivalent to the Back command. It types the previous message and sets the current message to be that message.

<= ^H 11e7

The <control-H> (or New-line) command is equivalent to the Back command. It types the previous message and sets the current message to be that message.

Other commands 11f

<= Verbose 11f1

This is a binary switch which causes the program to go into either 'Short typeout mode' or 'Long typeout mode', and tells you which is the setting that it changes to. The default is 'Short typeout mode'. Long typeout mode gives additional prompting regarding what is expected to be typed in.

<= Inclusion of length in header 11f2

This command is a binary switch which causes the program to go into a mode where header listings caused by the Header command will have the number of characters in the message included as part of the subject field. The default is that the length will not be included. Note that when you read a file initially, the length of 'recent' messages will always be included in the initial listing of recent headers.

<= ; <comment> 11f3

This command is mainly intended to allow you to talk with somebody over a link while you are in MSG. It eats all characters except <return> and control-Z (^Z), which return you to the command level of MSG. Two other characters have special effects. <delete> (<rub-out>) will type the string ' XXX ' and is useful in indicating that the previous word (or phrase) should be ignored. <line-feed> will cause effectively a carriage return and tab sequence to be typed. This way you can type more than one line of text. NOTE: the standard TENEX editing characters (e.g. control-A) are treated as any other character and perform no special function.

Command to Run Other Programs 11g

<= Sndmsg [confirm] 11g1

This command will start up SNDMSG and give control of the terminal to it. When SNDMSG is finished (i.e. when you have sent the message), it will turn control back to MSG in the same state as it was before you sent the message. Delete (rubout) will ask if you wish to abort. If you provide a positive confirmation, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<= Answer current message
Send response to <ANSWER SUB-COMMAND> 11g2

This facility allows you to send a message to the sender of the 'current' message, and (at your discretion) those people to whom the 'current' message was sent, without having to type their addresses to Sndmsg.

The <ANSWER SUB-COMMAND> can be any of the following:

S -- Sender of the original message only

O -- Original sender with a cc: to <login directory>

A -- All recipients of the original message (that is, the sender of the mesage and all addresses on the To: and cc: portions of the message)

<return> -- same as S.

Typing anything else aborts the command. The header of the current message is also typed so that you may be sure you are answering the correct message. If relevant, it will issue a warning if either the To: or cc: destination fields of the message have a destination list as part of the field (like LISP-USERS:). When control is given to you to type your answer, you will be typing to the message acquisition portion of SNDMSG (i.e. that part which prompts you by typing "Message (? for help):"). Delete (rubout) will ask if you wish to abort. If you give positive confirmation, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<= Forward (message sequence) <MSG-SEQUENCE> 11g3

This facility will allow you to send copies of messages you have received to other people. It will hand SNDMSG the subject and those messages you want forwarded, and leave you in SNDMSG in such a way that the message being forwarded can be edited, or your own comments added. You will be left in SNDMSG as though you had typed the

forwarded message in yourself, when done, type a control-Z and then specify, in the standard way, to whom the mail is going. Delete (rubout) will ask if you wish to abort. If you give a positive confirmation in the standard way, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<= Jump into lower fork running: <FILE-NAME> 11g4

This command is an escape in MSG in case you wish to run another program such as TECO, PUB, the EXEC, and so on. It searches directories to try to find the program you are asking it to run. The search list is, in order, <SUBSYS>, <SYSTEM>, your connected directory, and the login directory if different from the connected directory. This way, you can run EXEC without having to type the complete information (<SYSTEM>EXEC.SAV).

If you decide to leave the lower fork, but want to continue it at a later time, all you need do is type an escape as the first character of the file name you are requested to provide. This will cause the old file name (with an appropriate message) to be printed, and then you will be asked to confirm in the standard way. If you provide a positive confirmation, you will be asked it you want to continue or start that program. Typing 'C' for continue will put you back in the lower fork at the place where you exited; typing 'S' for start will restart the program.

<= "Exec [confirm] 11g5

When you type control-E, the program will type "Exec" to you and ask for confirmation. This command is intended to give you a new copy of the EXEC with a minimum of hassles. To leave that exec and return to MSG, type Quit. If you decide that you want a copy of the exec again, and you use this command, you will be given the same exec with all of your context intact.

This completes the list of MSG commands. There is only one item left to mention. 11h

Receiving New Messages while Using MSG 12

MSG, on typing a command or returning from the execution of a command, checks to see if your currently active message file, usually MESSAGE.TXT, has been written into. If it has, it prints out that fact and the headers for the new messages. It then executes your command or returns to command mode, accordingly. 12a

Command Summary 13

Cmnd, Char, Meaning 13a

A Answer current message
Send response to
S -- Sender of current message only
O -- Original sender, cc: to <login directory>
A -- All recipients of current message
<return> -- same as S 13b

B Backing up -- previous message is: 13c

" Same as Backing up 13d

"H Same as Backing up 13e

C Current message is nn of mm messages
in file: <FILE-NAME> 13f

D Delete (message sequence) <MSG-SEQUENCE> 13g

"E Exec [confirm] 13h

E Exit and update old file <FILE-NAME> [confirm] 13i

F Forward (message sequence) <MSG-SEQUENCE> 13j

G Go to message number: <number> 13k

H Headers (message sequence) <MSG-SEQUENCE> 13l

I Inclusion of length in header 13m

J Jump into lower fork running file: <program name> [confirm] 13n

L List (message sequence) <MSG-SEQUENCE>
on file name: <FILE-NAME> 13o

M Move (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME> 13p

N Next message is: 13q

<lf> (line feed) same as Next message is: 13r

O Overwrite old file <FILE-NAME> [confirm] 13s

P Put (message sequence) <MSG-SEQUENCE>
into file name: <FILE-NAME> 13t

Q Quit [confirm] 13u

R Read file name: <FILE-NAME> 13v

S Sndmsg [confirm] 13w

T Type (message sequence) <MSG-SEQUENCE> 13x

U Undelete (message sequence) <MSG-SEQUENCE> 13y

V Verbose -- provides more prompting 13z

W Write file <FILE-NAME> sorted by message arrival time
[confirm] 13a@

? ? Type command character for its description, ? for summary 13aa

; ; Comment -- <return> or ^Z returns you to command level 13ab

To abort a command, type rubout (delete). Abort terminal output
with ^O (control-O). Confirm with Y or <return>. 13ac

Errors while Reading a Message File 14

When reading a file in MSG (either at startup or with the 'Read'
command, the file MUST be in the so-called message file format.
If MSG recognizes that the file does NOT conform to this format,
you will be told so. The following are the circumstances which
might cause the file to become unreadable, and some suggestions
for getting around the problems. 14a

The file is a message file (that is, one or more valid
messages have been read from it), but somewhere in the middle
it does not conform to the message file format. It could be:
(1) It has a hole in it. Read the file with a text editor to
get rid of the hole, and write it back out, and reuse MSG. Try
this first. If this doesn't work, MSG will give you an error
at the same place. Then you can try the second suggestion: (2)
If suggestion 1 didn't work, then the file has internal byte
counts which do not match the actual file. Either you used a
text editor on your message file changing the number of bytes
but not the byte counts or your file was mysteriously altered.
The date of a message could not be read. Either the byte count
for the last message read was wrong, or there is junk between
the last message read and the one with the error. Using some
editor, find the last message read. The first line of that

message contains a date-and-time followed by a byte count indicating how many characters are in the message body starting on the following line. Skip that many characters of the message body. You should be at the date-and-time line of the next message. If there is junk there, delete it. Otherwise, try to fix the count so it is pointing at the date-and-time of the next message. 14a1

The beginning of the file does not conform to the message file format. It could be: (1) the file is not a message file -- sorry, we can't help you there. (2) It is a message file with a bad first line -- probably a blank line. Read the file with a text editor. If the second line begins with a time and date then delete the first line and reuse MSG on the new file. (3) It is a message file with a hole at the beginning. Read it with a text editor to get rid of the hole, write it out and reuse MSG. 14a2

JBP 29-APR-75 01:52 25806

'MSG' mail reading and processing program documentation

(J25806) 29-APR-75 01:52;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /SRI-ARC([INFO-ONLY]) NSW([INFO-ONLY]) ; Sub-collections: SRI-ARC NSW; Clerk: JBP; Origin: < POSTEL, MSG-MANUAL,NLS;4, >, 29-APR-75 01:48 JBP ;;;;####;

KIRK 29-APR-75 02:47 25807

Design for a Readmail Tool

For your review before going to the NSW contractors for review. To be output processed for offline reading. There is a sequential print file called <HELP>READMAIL.PRINT at BBNB for this purpose.

Design for a Readmail Tool

READMAIL

The Readmail tool provides a variety of commands to help an executive secretary manage online communications. These include reading, filing, forwarding, and answering online mail as well as a simple-minded calendar with an automatic daily reminder. This document begins with a general description of your online "desk top". This is followed by a description of each Readmail command. For information on how to gain access to tools, see "Introduction to the NSW". For information on how to use NLS tools in general, see "Getting into NLS". 1

INITIAL FILE

Your online mail is delivered to your "initial" file which is the file you see when you first go to the NLS Editor. The name of this file is made from your identifier which is usually your initials. See the Sendmail documentation to learn about identifiers. You should be in your initial file when you use Readmail. Your initial file functions as your online "desk top" and can be thought of as partitioned into various areas called categories. 1a

CATEGORIES

New mail is filed in the category named "new". Reading an item in this category moves it to the category named "old". Items that you have asked to be reminded of are located in the category named "calendar". Mail you have authored is located in your "author" category. Deleted mail is located in a "deleted" category until you request that it be undeleted or expunged. You can create any other categories or categories- within- categories that you like using the NLS Editor command "Insert". 1b

ITEMS

Some Readmail commands ask you for ITEMS; specify the number of a piece of mail, numbers of pieces of mail, or the name of a category. Each item within a category is given a number when you first view the category. The number appears at the beginning of the item. To specify a particular item, just type this number. The number stays with an item until you quit from Readmail. You can specify a whole category of ITEMS by typing the category name. 1c

% Designers comment: It will probably be possible to bug words for category names and item numbers wherever ITEMS is the parameter. 1c1

COMBINATIONS OF ITEMS

You can specify a group of items within the same category by placing a dash between the numbers of the first and last items thus: 1-10. Groups of groups can also be specified by separating the groups with spaces thus: 1-4 6-8 10. ITEMS in a different category can be specified by prefacing the number(s) with the name of the

category thus: author 1.
You can type filenames separated by a comma in front of or in place of category names. You can also specify a category within a category by typing the super category a space and the sub category. 1c2

JOURNALIZED ITEMS
To retrieve items recorded in the NLS Journal for which you know the item's Journal number, use the Brief or Verbose command and type the number followed by a comma, 1c3

ENVELOPE
Each piece of mail has certain elements of information located in places chosen for their optimum online use. The resulting format is called the "envelope". First of all there is a citation or header. Under the citation is the distribution list followed by various optional fields such as Comments, Date Received, Items updated or made obsolete, etc. The body of the piece of mail, if delivered, occupies the rest of the envelope. More about this later. Here is a sample envelope. 1d

3, GMT1332-22-OCT-75/AUTHOR: Title or subject of this piece of mail <;w> <J12345,;wg> [ACTION]
To: IDENTIFIERS OF PRIMARY RECIPIENTS
Cc: PEOPLE WHO RECEIVED COPIES
Comment: This is a sample envelope.
Received at: PDT1333 22-OCT-75
This is the body of the piece of mail. 1d1

% Design Comment % 1d2

This differs in one respect from the default format generated by RLL's "cooperative design". The time is in a form which allows it to be a statement name. This provides two capabilities. 1d2a

It provides a handle to specific items that works outside of Readmail and is independent of category. 1d2a1

It allows turning names off and getting more of the essential title information in clipped views. 1d2a2

It should also be useful for future automatic processes such as schedulers, diaries and sorters. 1d2b

Slash was chosen as a name delimiter to divide the date and author because it is unlikely to accidentally create conflicts with category names. The slash could have spaces around it if proves to be worth the two extra characters. 1d2c

CITATION

Design for a Readmail Tool

The beginning of every piece of mail consists of a citation containing the time and date the item was sent, the identifier of the author, and the title or subject. If the item has been recorded in the NLS Journal, a link to it's online location will be included. If there are any special notes such as "ACTION", "Private", and/or "Unrecorded", they will also be included in the citation. 1d3

TIME AND DATE GMT1332-22-OCT-75: a handle
Your time zone makes up the first three letters of the citation and follows the Readmail item number. It is followed immediately by the time of day the piece of mail was sent using a 24 hour clock with no punctuation. After the time of day comes the day, month, and year. The time and date are a handle with which you can specify any item independent of it's category. 1d3a

AUTHOR AND TITLE
The author's identifier is placed after the time. If there is more than one author, their identifiers are separated by spaces. A colon divides the author field and the title field. The title follows the author part and is terminated by a carriage return. 1d3b

LINKS
If the item has been recorded in the NLS journal, a link to it's location in the journal will be placed after the title. Links are surrounded by angle-brackets. See ... for more information about links. If the body of the item has been copied to your initial file, the short link: <:w> will appear before the link to the body. This is so your copy of the body of the item will be displayed rather than taking the time to go find it in the NLS journal. 1d3c

NOTES
If the sender has written an additional note regarding the item intended for you especially, it will appear in square brackets at the end of the citation part of the envelope. If the item has been sent "private", "Unrecorded", or for your "ACTION", these notes will also appear here. 1d3d

ELEMENTS
Immediately under the citation is the list of the identifiers of the primary recipients preceded by the word "To:". After that is the list of secondary recipients (if any) preceeded by the word "Cc:". Various extra elements of information may appear after the distribution list. For a complete list of these with explanations of their use, see the Sendmail commands. 1d4

BODY
The entire contents of some items are placed in your initial file at the end of their "envelopes". For others there will just be a

"link" in the citation to a file containing the actual information. When you ask to see one of these items, the message "Catalog file" will be typed while the files are being found before they are printed for you. Otherwise, the information will look as if it is located in your file. 1d5

1d6

Design for a Readmail Tool

COMMANDS IN READMAIL

The following command descriptions begin with the syntax of the command followed by a short description of the command. See ... for the meanings of syntax symbols and for general information on how to command tools integrated into the NLS environment. 1e

Answer (item number) CONTENT (message) CONTENT (send to all recipients?) ANSWER OK:

The Readmail command "Answer" allows you to type a message in response to an item and have it automatically sent to all of the recipients or just to the author(s). Specify the item number for the first CONTENT. A NULL item number defaults to the last item you read. Type in the message for the second CONTENT and terminate it with your OK button. If you terminate the command at this point by hitting another OK, your response will be sent to everyone in the "To" list including the author and copies will be sent to everyone in the "Cc" list. Alternatively, you can answer "No" when asked "send to all recipients?" and only send your response to the author(s) of the item. A copy of your response will appear in your personal "author" category. 1e1

Brief (view for) ITEMS OK:

The Readmail command "Brief" will show you the first line of each category or item one level under the category you specify for ITEMS. To see a list of your top-level categories, specify zero for ITEMS in the "Brief" command. A number is assigned to each item or category. CTRL-O stops printing. To read one of the items, use the Readmail commands "Interrogate", "Verbose", or "Next". 1e2

Category ITEMS OK:

The Readmail command "Category" puts you in the category you specify for ITEMS without having it printed. If you have just entered Readmail, your category is already set to be "NEW". If you use the Interrogate command or if you have just used the Brief or Verbose command, you do not need to use the "Category" command. 1e3

Copy (item(s)) ITEMS (to category) ITEMS OK:

You can file items under more than one category by using the Readmail command "Copy (item(s))". Specify the item you wish to copy for the first ITEMS. Specify the category under which you wish to file it for the second ITEMS. 1e4

Delete ITEMS OK:

The Readmail command "Delete" moves the ITEMS you specify to a

category named "deleted". Readmail has an "Undelete" command which moves ITEMS you have deleted to the category named "new". You can use the Readmail command "Expunge" to destroy the "deleted" category or deleted items can be expunged automatically when you quit Readmail. See Quit. 1e5

Execute (command in) TOOL: 1e6

% See ... for a description of the Goto command. 1e6a

Expunge (all deleted items) OK:
The Readmail command "Expunge" destroys your deleted items for good. This can happen automatically when you Quit Readmail. See Delete and Quit. The Undelete command will not undelete items that have been expunged. 1e7

ForWard (item number) ITEMS (for) ACT/INFO (to) CONTENT OK:
The Readmail command "ForWard" allows you to pass on the ITEMS you specify to the identlist you specify for CONTENT. ACT/INFO wants either the command-word Action or Information. If you specify a journal number, Readmail will first look to see if you have any items with the same number. If not, then the journal item with that number will be forwarded. 1e8

Goto TOOL OK: 1e9

% See ... for a description of the Goto command. 1e9a

Include QUALITY CONTENT OK:
The Readmail command "Include" allows you to view only items with certain qualities. For QUALITy, specify the command-word "Authors", "Title words" or "Dates". A QUALITY filter remains in effect until you re-specify it or use the corresponding Readmail "Omit" command which works as the inverse of the "Include" command. Those in terse recognition mode must type a space before they can use the Include or Omit commands.
QUALITY = Authors / Title words / Dates. 1e10

Include Authors CONTENT OK:
specify for CONTENT the IDENTLIST of authors you wish to see.

NULL for CONTENT means "ALL". In the Omit Authors command, NULL for CONTENT means "NONE". 1e10a

Include Title words CONTENT OK:
Specify for CONTENT some words in the titles of the items you wish to see. NULL for CONTENT means "ALL". In the Omit Title words command, NULL for CONTENT means "NONE". 1e10b

Include Dates (from) CONTENT (to) CONTENT OK:
Specify for each CONTENT the range of dates (with times if desired) of the items you wish to see. NULL in the "from" field specifies the beginning of time, NULL in the "to" field specifies right now. 1e10c

Interrogate OK:
After typing an OK, the Readmail command "Interrogate" first asks for the name of the category you wish to examine. When you first enter Readmail your category is automatically set to "NEW". If this is the category you want, you can specify NULL for the category. Interrogate then shows you the first item in the category and asks if you want to answer it. If you do, it allows you to type a message. Then it asks if you want to be reminded of it. If you do, it allows you to enter the day to be reminded. Then it asks if you want to file it and if so, where. If you do not want to file it, Interrogate asks if you want to delete it. When you are done, the next item is shown. After the last item has been shown and the interrogation answered, you are asked if you want to quit Readmail. To stop the interrogation at any point, type your Command Delete key. 1e11

Move (item number:) ITEMS (under category) ITEMS OK
Use the Readmail command "Move" to file the item with the number you specify for the first ITEMS in the category you specify for the second ITEMS. See also: Copy. 1e12

Next (item) OK:
Use the Readmail command "Next" to see a full view of the next item in your current category. Typing the line feed key also shows you the Next item. 1e13

Omit QUALITY CONTENT OK:
The Readmail command "Omit" works as the opposite of the Readmail command "Include". 1e14

Design for a Readmail Tool

Output ITEMS (to printer) OK:
The Readmail command "Output" prints what you indicate for ITEMS at the hard copy device specified for you in your profile. If you specify the name of a category with more than one item, a brief view of that category will appear on the first page followed by a full view of each item in that category. If you specify NULL for ITEMS, your current item will be printed. 1e15

Quit (update file?) Y/N:
The Readmail command "Quit" returns you to your previous tool. If you type y or your OK, then items you have deleted will be expunged and your file will be updated. See Delete. 1e16

Remind (me of) Item(s) ITEMS (on day) CONTENT OK:
The Readmail command "Remind" will move the ITEMS you specify to the category named "calendar". The first time you use Readmail after the date and time you specify for CONTENT, the ITEMS will reappear in your "NEW" category. Specify the date and time in the following form DD/MM/YY HH:MM. If you leave off the time, 00:00 (midnight) will be assumed. If you leave off the DD/MM/YY, the current day will be assumed. If you specify NULL for CONTENT, the item will be moved immediately to (or left in) the category "new". 1e17

Remind (me of) Note CONTENT (on day) CONTENT OK:
Instead of "Item(s)" you can specify "Note" and type in or point to any personal information you wish to be reminded of. 1e17a

Sort (category) OK/ORDER:
The Readmail command "Sort" orders the items in your current category. OK causes a sort by date and time, most recent at the top. Alternatively, you can specify for ORDER "Oldest (first) OK" or "Authors OK". If you specify "Authors", Those items with the same first author will be grouped together. 1e18

Undelete ITEMS OK:
The Readmail command "Undelete" causes the deleted item you specify to be moved from the "deleted" category to your current category. See Delete. 1e19

Verbose (view for) ITEMS OK:
The Readmail command "Verbose" will show you a full view of the ITEMS you specify. If you specify a category name for ITEMS, a full view of each item in that category will be shown at your terminal. A

number is assigned to each item. <CTRL-O> stops printing. See also: Output. 1e20

REPEAT: 1e21

% See REPEAT in the Help command for an explanation of this function. 1e21a

LF: line feed 1e22

% See the Readmail "Next" command. 1e22a

LESS-THAN: go back
Pushing the less-than key < will cause you to return to the last place you viewed using Readmail. 1e23

" go up
Pushing " will cause you to go up to see, for example, the entire category you are in after looking at one of its items. 1e24

% Somehow there should be a way of showing MESSAGE.TXT messages either by automatically moving them into nls whenever the READMAIL subsystem is fired up or else having a command such as "Read Sndmsg". 1e25

% We also need a "Read U.S. postal mail" command which asks the secretary questions towards entering a letter as an XDOC item. The XDOC procedures for handling off-line items classified in the online medium would be part of the Readmail help description file. 1e26

1e27

KIRK 29-APR-75 02:47 25807

Design for a Readmail Tool

(J25807) 29-APR-75 02:47;;; Title: Author(s): KirK E. Kelley/KIRK;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: KIRK;

Readmail Design

< MJOURNAL, 25807.NLS;1, >, 29-APR-75 18:00 XXX ;;;; .HJOURNAL="KIRK 29-APR-75 02:47 25807"; Title: .H1="Design for a Readmail Tool"; Author(s): Kirk E. Kelley/KIRK; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: KIRK; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

.PEL; .PN=PN-1; .GCR;For your review before going to the NSW contractors for review. To be output processed for offline reading. There is a sequential print file called <HELP>READMAIL.PRINT at BBNB for this purpose.

READMAIL.Center;.GCR;.LBS=1;.BLM=-6;

The Readmail tool provides a variety of commands to help an executive secretary manage online communications. These include reading, filing, forwarding, and answering online mail as well as a simple-minded calendar with an automatic daily reminder. This document begins with a general description of your online "desk top". This is followed by a description of each Readmail command. For information on how to gain access to tools, see "Introduction to the NSW". For information on how to use NLS tools in general, see "Getting into NLS".

INITIAL FILE

Your online mail is delivered to your "initial" file which is the file you see when you first go to the NLS Editor. The name of this file is made from your identifier which is usually your initials. See the Sendmail documentation to learn about identifiers. You should be in your initial file when you use Readmail. Your initial file functions as your online "desk top" and can be thought of as partitioned into various areas called categories.

CATEGORIES

New mail is filed in the category named "new". Reading an item in this category moves it to the category named "old". Items that you have asked to be reminded of are located in the category named "calendar". Mail you have authored is located in your "author" category. Deleted mail is located in a "deleted" category until you request that it be undeleted or expunged. You can create any other categories or categories- within- categories that you like using the NLS Editor command "Insert".

ITEMS

Some Readmail commands ask you for ITEMS; specify the number of a piece of mail, numbers of pieces of mail, or the name of a category. Each item within a category is given a number when you first view the category. The number appears at the beginning of the item. To specify a particular item, just type this number. The number stays with an item until you quit from Readmail. You can specify a whole category of ITEMS by typing the category name.

% Designers comment: It will probably be possible to bug words for category names and item numbers wherever ITEMS is the parameter.

COMBINATIONS OF ITEMS

You can specify a group of items within the same category by placing a dash between the numbers of the first and last items

thus: 1-10. Groups of groups can also be specified by separating the groups with spaces thus: 1-4 6-8 10. ITEMS in a different category can be specified by prefacing the number(s) with the name of the category thus: author 1.

You can type filenames separated by a comma in front of or in place of category names. You can also specify a category within a category by typing the super category a space and the sub category.

JOURNALIZED ITEMS

To retrieve items recorded in the NLS Journal for which you know the item's Journal number, use the Brief or Verbose command and type the number followed by a comma.

ENVELOPE

Each piece of mail has certain elements of information located in places chosen for their optimum online use. The resulting format is called the "envelope". First of all there is a citation or header. Under the citation is the distribution list followed by various optional fields such as Comments, Date Received, Items updated or made obsolete, etc. The body of the piece of mail, if delivered, occupies the rest of the envelope. More about this later. Here is a sample envelope.

3. GMT1332-22-OCT-75/AUTHOR: Title or subject of this piece of mail
<:w> <J12345,:wg> [ACTION]
To: IDENTIFIERS OF PRIMARY RECIPIENTS
Cc: PEOPLE WHO RECEIVED COPIES
Comment: This is a sample envelope.
Received at: PDT1333 22-OCT-75
This is the body of the piece of mail.

% Design Comment %

This differs in one respect from the default format generated by RLL's "cooperative design". The time is in a form which allows it to be a statement name. This provides two capabilities.

It provides a handle to specific items that works outside of Readmail and is independent of category.

It allows turning names off and getting more of the essential title information in clipped views.

It should also be useful for future automatic processes such as schedulers, diaries and sorters.

Slash was chosen as a name delimiter to divide the date and author because it is unlikely to accidentally create conflicts with category names. The slash could have spaces around it if proves to be worth the two extra characters.

CITATION

The beginning of every piece of mail consists of a citation

containing the time and date the item was sent, the identifier of the author, and the title or subject. If the item has been recorded in the NLS Journal, a link to it's online location will be included. If there are any special notes such as "ACTION", "Private", and/or "Unrecorded", they will also be included in the citation.

TIME AND DATE GMT1332-22-OCT-75: a handle
Your time zone makes up the first three letters of the citation and follows the Readmail item number. It is followed immediately by the time of day the piece of mail was sent using a 24 hour clock with no punctuation. After the time of day comes the day, month, and year. The time and date are a handle with which you can specify any item independent of it's category.

AUTHOR AND TITLE
The author's identifier is placed after the time. If there is more than one author, their identifiers are separated by spaces. A colon divides the author field and the title field. The title follows the author part and is terminated by a carriage return.

LINKS
If the item has been recorded in the NLS journal, a link to it's location in the journal will be placed after the title. Links are surrounded by angle-brackets. See ... for more information about links. If the body of the item has been copied to your initial file, the short link: <:w> will appear before the link to the body. This is so your copy of the body of the item will be displayed rather than taking the time to go find it in the NLS journal.

NOTES
If the sender has written an additional note regarding the item intended for you especially, it will appear in square brackets at the end of the citation part of the envelope. If the item has been sent "Private", "Unrecorded", or for your "ACTION", these notes will also appear here.

ELEMENTS
Immediately under the citation is the list of the identifiers of the primary recipients preceded by the word "To:". After that is the list of secondary recipients (if any) preceeded by the word "Cc:". Various extra elements of information may appear after the distribution list. For a complete list of these with explanations of their use, see the Sendmail commands.

BODY
The entire contents of some items are placed in your initial file at the end of their "envelopes". For others there will just be a "link" in the citation to a file containing the actual information. When you ask to see one of these items, the message "Catalog file" will be typed while the files are being found before they are printed for you. Otherwise, the information will look as if it is located in your file.

.PES;

COMMANDS IN READMAIL

The following command descriptions begin with the syntax of the command followed by a short description of the command. See ... for the meanings of syntax symbols and for general information on how to command tools integrated into the NLS environment. .LBS=2;

Answer (item number) CONTENT (message) CONTENT
(send to all recipients?) ANSWER OK:

The Readmail command "Answer" allows you to type a message in response to an item and have it automatically sent to all of the recipients or just to the author(s). Specify the item number for the first CONTENT. A NULL item number defaults to the last item you read. Type in the message for the second CONTENT and terminate it with your OK button. If you terminate the command at this point by hitting another OK, your response will be sent to everyone in the "To" list including the author and copies will be sent to everyone in the "Cc" list. Alternatively, you can answer "No" when asked "send to all recipients?" and only send your response to the author(s) of the item. A copy of your response will appear in your personal "author" category.

Brief (view for) ITEMS OK:

The Readmail command "Brief" will show you the first line of each category or item one level under the category you specify for ITEMS. To see a list of your top-level categories, specify zero for ITEMS in the "Brief" command. A number is assigned to each item or category. CTRL-O stops printing. To read one of the items, use the Readmail commands "Interrogate", "Verbose", or "Next".

Category ITEMS OK:

The Readmail command "Category" puts you in the category you specify for ITEMS without having it printed. If you have just entered Readmail, your category is already set to be "NEW". If you use the Interrogate command or if you have just used the Brief or Verbose command, you do not need to use the "Category" command.

Copy (item(s)) ITEMS (to category) ITEMS OK:

You can file items under more than one category by using the Readmail command "Copy (item(s))". Specify the item you wish to copy for the first ITEMS. Specify the category under which you wish to file it for the second ITEMS.

Delete ITEMS OK:

The Readmail command "Delete" moves the ITEMS you specify to a category named "deleted". Readmail has an "Undelete" command which moves ITEMS you have deleted to the category named "new". You can use the Readmail command "Expunge" to destroy the "deleted" category or deleted items can be expunged automatically when you quit Readmail. See Quit.

Execute (command in) TOOL: .IGRest;

##<fe-wm !execute>##

% See ... for a description of the Goto command.

Expunge (all deleted items) OK:
The Readmail command "Expunge" destroys your deleted items for good. This can happen automatically when you Quit Readmail. See Delete and Quit. The Undelete command will not undelete items that have been expunged.

Forward (item number) ITEMS (for) ACT/INFO (to) CONTENT OK:
The Readmail command "Forward" allows you to pass on the ITEMS you specify to the identlist you specify for CONTENT. ACT/INFO wants either the command-word Action or Information. If you specify a journal number, Readmail will first look to see if you have any items with the same number. If not, then the journal item with that number will be forwarded.

Goto TOOL OK: .IgRest;
##<fewm, goto>##

% See ... for a description of the Goto command.

Include QUALITY CONTENT OK:
The Readmail command "Include" allows you to view only items with certain qualities. For QUALITY, specify the command-word "Authors", "Title words" or "Dates". A QUALITY filter remains in effect until you re-specify it or use the corresponding Readmail "Omit" command which works as the inverse of the "Include" command. Those in terse recognition mode must type a space before they can use the Include or Omit commands.
QUALITY = Authors / Title words / Dates.

Include Authors CONTENT OK:
Specify for CONTENT the IDENTLIST of authors you wish to see. NULL for CONTENT means "ALL". In the Omit Authors command, NULL for CONTENT means "NONE".

Include Title words CONTENT OK:
Specify for CONTENT some words in the titles of the items you wish to see. NULL for CONTENT means "ALL". In the Omit Title words command, NULL for CONTENT means "NONE".

Include Dates (from) CONTENT (to) CONTENT OK:
Specify for each CONTENT the range of dates (with times if desired) of the items you wish to see. NULL in the "from" field specifies the beginning of time. NULL in the "to" field specifies right now.

Interrogate OK:
After typing an OK, the Readmail command "Interrogate" first asks for the name of the category you wish to examine. When you first enter Readmail your category is automatically set to "NEW". If this is the category you want, you can specify NULL for the category. Interrogate then shows you the first item in the category and asks if you want to answer it. If you do, it allows

you to type a message. Then it asks if you want to be reminded of it. If you do, it allows you to enter the day to be reminded. Then it asks if you want to file it and if so, where. If you do not want to file it, Interrogate asks if you want to delete it. When you are done, the next item is shown. After the last item has been shown and the interrogation answered, you are asked if you want to quit Readmail. To stop the interrogation at any point, type your Command Delete key.

Move (item number:) ITEMS (under category) ITEMS OK
Use the Readmail command "Move" to file the item with the number you specify for the first ITEMS in the category you specify for the second ITEMS. See also: Copy.

Next (item) OK:
Use the Readmail command "Next" to see a full view of the next item in your current category. Typing the line feed key also shows you the Next item.

Omit QUALITY CONTENT OK:
The Readmail command "Omit" works as the opposite of the Readmail command "Include".

Output ITEMS (to printer) OK:
The Readmail command "Output" prints what you indicate for ITEMS at the hard copy device specified for you in your profile. If you specify the name of a category with more than one item, a brief view of that category will appear on the first page followed by a full view of each item in that category. If you specify NULL for ITEMS, your current item will be printed.

Quit (update file?) Y/N:
The Readmail command "Quit" returns you to your previous tool. If you type y or your OK, then items you have deleted will be expunged and your file will be updated. See Delete.

Remind (me of) Item(s) ITEMS (on day) CONTENT OK:
The Readmail command "Remind" will move the ITEMS you specify to the category named "calendar". The first time you use Readmail after the date and time you specify for CONTENT, the ITEMS will reappear in your "NEW" category. Specify the date and time in the following form DD/MM/YY HH:MM. If you leave off the time, 00:00 (midnight) will be assumed. If you leave off the DD/MM/YY, the current day will be assumed. If you specify NULL for CONTENT, the item will be moved immediately to (or left in) the category "new".

Remind (me of) Note CONTENT (on day) CONTENT OK:
Instead of "Item(s)" you can specify "Note" and type in or point to any personal information you wish to be reminded of.

Sort (category) OK/ORDER:
The Readmail command "Sort" orders the items in your current category. OK causes a sort by date and time, most recent at the top. Alternatively, you can specify for ORDER "Oldest (first) OK" or "Authors OK". If you specify "Authors", Those items with

the same first author will be grouped together.

Undelete ITEMS OK:
The Readmail command "Undelete" causes the deleted item you specify to be moved from the "deleted" category to your current category. See Delete.

Verbose (view for) ITEMS OK:
The Readmail command "Verbose" will show you a full view of the ITEMS you specify. If you specify a category name for ITEMS, a full view of each item in that category will be shown at your terminal. A number is assigned to each item. <CTRL-O> stops printing. See also: Output.

REPEAT: .IgRest;
##<nlsum, repeat>##

% See REPEAT in the Help command for an explanation of this function.

LF: line feed .IgRest;
##<next>##

% See the Readmail "Next" command.

LESS-THAN: go back
Pushing the less-than key < will cause you to return to the last place you viewed using Readmail.

^ go up
Pushing ^ will cause you to go up to see, for example, the entire category you are in after looking at one of its items.

% Somehow there should be a way of showing MESSAGE.TXT messages either by automatically moving them into nls whenever the READMAIL subsystem is fired up or else having a command such as "Read Sndmsg".

% We also need a "Read U.S. Postal mail" command which asks the secretary questions towards entering a letter as an XDOC item. The XDOC procedures for handling off-line items classified in the online medium would be part of the Readmail help description file.

.PES;

KIRK 29-APR-75 02:53 25808

Just happened to notice I'm a member

Distributed to THRG

KIRK 29-APR-75 02:53 25808

Just happened to notice I'm a member

Ra3y, what is THRG for?

1

KIRK 29-APR-75 02:53 25808

Just happened to notice I'm a member

(J25808) 29-APR-75 02:53;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /THRG([INFO-ONLY]) ; Sub-Collections: SRI-ARC THRG;
Clerk: KIRK;

EKM 30-APR-75 15:53 25809

Tabs: Reponse to (MJOURNAL,25794,)

While your suggestion for implementing tabs with backspaces (MJOURNAL,25794,) has some merit we can't do it that way for the following reasons: 1

It would require changes to every formatter we have 1a

The output processor 1a1

Output quickprint 1a2

The display formatters for Line Processors and Imlacs 1a3

etc. 1a4

An unknown number of FIND statements in NLS and user programs would no longer work as expected 1b

It would not work on TI terminals or other terminals which do not allow moving the carriage back. Since we are putting in this "whistle" for people who use TI terminals this would be a little silly. 1c

We do not have time before July 1 to make the changes to code that would be required. 2

Therefore we should use your first implementation of putting spaces in instead of tabs. 3

Dick has given this little feature top priority. We must implement it with as little disruption to other work as possible. 4

EKM 30-APR-75 15:53 25809

Tabs: Reponse to (MJOURNAL,25794,)

(J25809) 30-APR-75 15:53;;; Title: Author(s): Elizabeth K. Michael/EKM; Distribution: /KIRK([ACTION]) RWW([INFO-ONLY]) JLE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: EKM; Origin: < MICHAEL, TABMESS.NLS;2, >, 30-APR-75 15:45 EKM ;;;;####;

RLL 30-APR-75 17:54 25810

bug: unnecessary refreshing of screen for substitute command

Apparently the substitute command refreshes the entire screen independent of the window you are substituting in and whether or not the screen needs a refresh for the update. With the poor response andother problems , I would think such bugs , if fixed , could save much time.

1

RLL 30-APR-75 17:54 25810

bug: unnecessary refreshing of screen for substitute command

(J25810) 30-APR-75 17:54;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /FEED([ACTION]) JCN([INFO-ONLY]) JHB([INFO-ONLY]) NDM([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

POOH 30-APR-75 18:50 25811

Documentation for Useroptions Tool in NSW

I am beginning to write a scenario that will explain to a programmer at Gunter the best way to set useroptions. Can you supply me with some information about the optimal useroptions setting for a programmer at Gunter who will be using the NLS 8.5 tool in NSW. 1

Documentation for Useroptions Tool in NSW

(J25811) 30-APR-75 18:50;;;; Title: Author(s): Ann Weinberg/POOH; Distribution: /LAC([ACTION]) KS([INFO-ONLY]) DVN([INFO-ONLY]) BBV([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: POOH;

BEV 30-APR-75 19:58 25812

Estimating Person-Loading for Documentation (9-Month Proposal)

Dick--Dirk and I discussed our person-loading further this afternoon and came up with a picture somewhat different from that which we discussed with you this morning. As a result, I am sending you this file, which lists our new estimate. Would appreciate your comments. Thanks. Bev

Estimating Person-Loading for Documentation (9-Month Proposal)

The following is our estimate of required person power for documentation for the 9-month proposal period as of 4/30/75. 1

TASKS	PERSON-MONTH	
Frontend	5.5	1b
Protocols	3	1c
Tools	24	1d
Edit. Support	1	1e
TOTAL	33.5	1f

These estimates are based on the following assumptions: FE (system-.5 pm/editing; manual-1 pm/editing; user-3pm/writing; report writing-1pm); Protocols (TBH-1pm/edit, interface; Design-1pm/edit; report writing-1pm); Tools (NLS doc.-9 pm/writing; Sp. user h.-2pm/write; User doc.-12pm/write; report writing-1pm). 1f1

Our estimate of person-months available is 33.5 (3.75 people X 9=33.5). This happens to come out very neat, as long as none of us spend any time whatsoever in overhead. As we see it now, there will not be any time to "sell " to applications. 2

Estimating Person-Loading for Documentation (9-Month Proposal)

(J25812) 30-APR-75 19:58;;; Title: Author(s): Beverly Boll/BEV;
Distribution: /RWW([ACTION]) &DIRT([INFO-ONLY]) ;
Sub-Collections: SRI-ARC DIRT; Clerk: BEV; Origin: < BOLI,
DOCPERSONPOWER,NLS;2, >, 30-APR-75 19:46 BEV ;;;####;

cml extensions

as we gain experience with the new CML, it is becoming clear that changes of the following nature are either needed or extremely nice to have. CML writers can assume the existence of the following capabilities (however they may not exist in this exact syntax): 1

provide an exitloop statement to get out of the looping construct without requiring the user to hit CommandDelete: 1a

exitloop := "EXIT" "LOOP" 1a1

control then passes to the next syntactical element following the loop statement. 1a1a

provide a syntax for referencing list elements: 1b

elementreference := listname '< elementindex '> 1b1

e.g. xxx<3> would reference the third element of list xxx; yyy<2><3> would reference the third element of the second element (itself a list) of list yyy. 1b1a

upgrade the IF capability as follows: 1c

if := "IF" ["NOT"] var [condclause] 1c1

condclause := ('< / "<=" / '= / '# / ">=" / '>) (var / value) 1c2

(note: it might be nice to have a CASE statement similar to the L10 CASE statement but this does not appear necessary) 1c3

beef up the procedure call syntax to be as follows: 1d

proccall := [var ('_ / ":_")] pname [rspecs] '([args] [': results] ') 1d1

rspecs := '[[inorout] [treturn] '] 1d2

inorout := "IN LINE" / "OUT OF LINE " / rulename 1d3

note that specifying a rulename implies an "OUT OF LINE" call 1d3a

treturn := '; rulename 1d4

this is the name of a rule to be executed in the case that an execution function does a PCP temporary return, this rule should terminate with a tempreturn statement (see below). 1d4a

NOTE: if no treturn rule is specified, the CLI will attempt to act rasonably for certain standard temporary returns, e.g, a HELP return which indicates recollection of a parameter from the user. if a treturn rule is specified that does not terminate with a tempreturn, then the rule will be executed before the CLI attempts to deal with the temporary return. 1d4b

add a tempreturn statement to return from temporary returns: 1e

tempreturn := "RESUME" '([args] ') / "ABORT" 1e1

provide the builtin variable "subreturntype", to be associated with every execution function call (either in or out of line), whose value is set and can be tested when an execution function does either a temporary or permanent return, the value of this variable can be either: 1f

for permanent returns: SUCCESS / FAILURE / ABORTED 1f1

for temporary returns: GENERALCOROUTINE / NOTE / HELP 1f2

cml extensions

(J25813) 30-APR-75 20:33;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: KEV; Origin: < SRIFE, CML-EXTENSIONS.NLS;1, >, 30-APR-75 17:21 KEV ;;;;####;

BEV 30-APR-75 20:34 25814

Notes on Primer

Ann--These are the notes that I mentioned to you yesterday, We can talk about them when we get together this a.m.

BEV 30-APR-75 20:34 25814

Notes on Primer

In reviewing the TNLS Prime the first time, I would suggest some sort of revisions in accordance with the suggestions below* 1

Why not begin the primer with a quick lesson on how to read one's mail? This would have been very helpful to me the first couple of weeks I was learning the system. 1a

My understanding is that Steps 1 and 2, as well as the first paragraph under INSTRUCTION on the first page, all need to be redone for NSW. 1b

Some of the <CTRL>-ch should be listed at the very beginning of the Primer. I am thinking particularly of <CTRL>-a, x, and o (and whatever does delete word). If the new user knew these controls from the beginning, it would save a lot of frustration, and give her a better sense of mastery (i.e., not a victim of the machine that does things she doesn't want it to do). 1c

Insert a command scenario for the second part of step 12 (Insert Statement). 1d

Suggestions from Kirk and Dirk: 2

Dirk--Readmail is covered under a separate document in readmail documentation. Yess on <CTRL-ch> and scenario for Step 12. 2a

Kirk--We need one overall introductory primer to nls in which logging in and basic control characters would be listed, along with other vital information on how to manipulate the system on a very basic level. Then on top of that we'd have the secretarial functions guide, and other guides, etc. The Secretarial Guide would basically consist of several primer/scenarios. In this basic introductory guide would go stuff now in the two-pager? Stuff from Charles, definitely. The Two-pager is an issue still to be resolved. 2b

Bev--Check out user documentation for readmail/sendmail. Does it in fact exist? 3

BEV 30-APR-75 20:34 25814

Notes on Primer

(J25814) 30-APR-75 20:34;;; Title: Author(s): Beverly Boli/BEV;
Distribution: /POOH([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
BEV; Origin: < BOLI, PRIMER.NLS;4, >, 30-APR-75 12:37 BEV
;;;;####;

list runtime program code

```
% Internal Format %                                                    1

   % The internal PDP-10 format of a list is:                          1a

      storwd                                                           1a1

      list: XWD M,,L                                                   1a2

      ledi (list element descriptor)                                   1a3

         ...                                                           1a3a

      ledM %                                                           1a4

   % Where storwd is a runtime package word that is used to indicate
   if (and where) the list resides in allocated storage. A value of
   zero means that the list has not been referenced. In that case M
   designates the number of (following) words that may be used for
   elements. %                                                         1b

% Definitions %                                                       2

   (ledr) RECORD % PDP-10 list element descriptor%                     2a

      ledval [18], %address/value of element%                         2a1

      ledtyp  [9], %element type%                                      2a2

      ledimd  [1], %ledval contains value if TRUE%                     2a3

      ledalo  [1]; %element space allocated if TRUE%                   2a4

   DECLARE EXTERNAL                                                    2b

      ldescr = 0 , % DESCRIPTOR %                                      2b1

      lnull  = 1 , % NULL %                                            2b2

      linteg = 2 , % INTEGER %                                         2b3

      lstrin = 3 , % STRING %                                          2b4

      llist  = 4 , % LIST %                                            2b5

      lnewde = 5 , % NEWDESCRIPTOR %                                   2b6

      lblock = 6 ; % BLOCK %                                           2b7

   % nulldescr = 001001000000B %                                       2c
```

```
    % ledval = 0 %                                                    2c1
    % ledtyp = 1 %                                                    2c2
    % ledimd = 1 %                                                    2c3
    % ledalc = 0 %                                                    2c4
(nulist) PROCEDURE (list REF, sublist, e1, e2) ;                      3
  % argument / result types %                                         3a
    % list - ADDRESS %                                               3a1
    % sublist - BOOLEAN %                                            3a2
    % e1 - INTEGER %                                                 3a3
    % e2 - INTEGER %                                                 3a4
  % If "sublist" is FALSE, effectively sets list.L to zero,
  Otherwise, discards elements "e1" through "e2", shifing forward
  any elements behind them. %                                         3b
  % declarations %                                                   3c
    LOCAL top, bot ;                                                  3c1
      % top - the highest numbered list element to delete %          3c1a
      % bot - the lowest numbered list element to delete %           3c1b
  IF sublist THEN bot _ e1 ELSE bot _ 1 ;                             3d
  IF sublist THEN top _ e2 ELSE top _ list.L ;                        3e
  % free all allocated storage in the elements to be eliminated %    3f
  FOR i _ bot UP UNTIL > top DO                                       3g
    freelm (list, i) ;                                                3g1
  % move any elements following the eliminated elements down %       3h
  WHILE top < list.L DO                                               3i
    BEGIN                                                             3i1
      list[bot] _ list[top+1] ;                                       3i1a
```

```
        BUMP bot ;                                                    3i1b
        BUMP top ;                                                    3i1c
     END ;                                                            3i2
  list.L _ bot - 1 ;                                                  3j
  return ;                                                            3K
(rdlelm) PROCEDURE (list REF, delete, index % => descr, value % ) ;   4
  % argument / result types %                                         4a
     % list - ADDRESS %                                              4a1
     % delete - BOOLEAN %                                            4a2
     % index - INTEGER %                                             4a3
     % descr - WORD %                                                4a4
     % value - INTEGER / ADDRESS %                                   4a5
  % Returns the descriptor "descr" for and value "value" (addr of
  L10 string, list, or block; or integer) of element "index" of list
  "list".  If "delete" is TRUE, the source element is replaced with
  a null descriptor. %                                                4b
  % declarations %                                                   4c
     LOCAL descr, value ;                                           4c1
  descr _ list[index] ;                                             4d
  IF list[index].ledtyp = linteg AND list[index].ledimd = FALSE THEN 4e
     value _ [list[index].ledval] ;                                 4e1
  ELSE                                                              4f
     value _ list[index].ledval ;                                   4f1
  IF delete THEN                                                    4g
     BEGIN                                                          4g1
        freelm ( list, index);                                      4g1a
```

```
list runtime program code


      END;                                                          4g2

      return ( descr, value );                                      4g3

(wrlelm) PROCEDURE (list REF, type, value, replace, index)          5

   % argument / result types %                                      5a

      % list - ADDRESS %                                           5a1

      % type - INTEGER %                                           5a2

      % value - INTEGER / ADDRESS / WORD %                         5a3

      % replace - BOOLEAN %                                        5a4

      % index - INTEGER %                                          5a5

   % If "replace" is FALSE, appends element of type "type" and value
   "value" to list "list", Otherwise, replaces element "index" with
   it. %                                                            5b

   IF NOT replace THEN                                              5c

      BEGIN                                                         5c1

         IF list.L = list.M THEN "go blooie" ;                      5c1a

         BUMP list.L ;                                              5c1b

         index _ list.L ;                                           5c1c

      END ;                                                         5c2

   freelm (list, index) ;                                           5d

   list[index] _ makelm ( type, value ) ;                           5e

return ;                                                            6

% BLC () %                                                         7

   % Initializes for list construction. %                          7a

% APLELM (type, value) %                                           8

   % Appends element of type "type" and value "value" to list in list
   construction workspace. %                                        8a
```

```
% APSUBL (32, list, e1, e2) %                                          9

   % If bit 35 of "flags" is on, appends all elements of list "list"
   to list in list construction workspace.  Otherwise, appends only
   elements "e1" through "e2".  If bit 34 of "flags" is on, the
   source descriptors are copied into the destination.  Otherwise,
   the source values are used, and typed as INTEGER at the
   destination.  If bit 33 of "flags" is on, the source elements are
   replaced with null descriptors. %                                   9a

   % flags %                                                           9b

      % bit 35 %                                                       9b1

         % 0 = append only elements e1 through e2 of list to the
         worklist %                                                    9b1a

         % 1 = append all elements of list to the worklist             9b1b

      % bit 34 %                                                       9b2

         % 0 = %                                                       9b2a

         % 1 = move %                                                  9b2b

      % bit 33 %                                                       9b3

         % 0 = %                                                       9b3a

         % 1 = copy                                                    9b3b

      % bit 32 %                                                       9b4

         % 0 = %                                                       9b4a

         % 1 = values are treated as integers %                        9b4b

% ELC (list => worklist) %                                            10

   % If argument "list" is present (i.e. non-zero), stores list in
   list construction workspace as list "list" and releases the
   workspace.  Otherwise, simply returns addr "worklist" of and
   responsibility for list in workspace. %                            10a

(makelm) PROCEDURE ( type, value % => descr % ) ;                     11

   % argument / result types %                                        11a

      % type = INTEGER %                                             11a1
```

list runtime program code

```
    % value = INTEGER / ADDRESS / WORD %                                    11a2

    % descr = WORD %                                                        11a3

  % makelm makes a descriptor for a new list element, including
  allocating storage and copying data of initial values. %                  11b

  CASE type OF                                                              11c

    = lnewde:                                                               11c1

      IF ckdesc ( value )                                                   11c1a

        THEN descr _ value                                                  11c1a1

        ELSE err ( s"list runtime package - makelm :
        newdescriptor " ) ;                                                 11c1a2

    = lnull:                                                                11c2

      descr _ nulldescr  ;                                                  11c2a

    = ldescr:                                                               11c3

      BEGIN                                                                 11c3a

        descr _ makelm ( [value].ledtyp, [value].ledval ) ;                 11c3a1

      END ;                                                                 11c3b

    = lintge:                                                               11c4

      IF value > 2^18 - 1 OR value < 0                                      11c4a

        THEN                                                                11c4a1

          BEGIN                                                             11c4a1a

            size _ 1 ;                                                      11c4a1a1

            blkadr _ getblk ( size, zone ) ;                                11c4a1a2

            blkadr[bhl] _ value ;                                           11c4a1a3

            descr.ledval _ &blkadr + bhl ;                                  11c4a1a4

            descr.ledtyp _ lintge ;                                         11c4a1a5

            descr.ledimd _ FALSE ;                                          11c4a1a6
```

list runtime program code

```
            descr.ledalo _ TRUE ;                                   11c4a1a7
          END                                                       11c4a1b
        ELSE                                                        11c4a2
          BEGIN                                                     11c4a2a
            descr.ledval _ value ;                                  11c4a2a1
            descr.ledtyp _ lintge ;                                 11c4a2a2
            descr.ledimd _ TRUE ;                                   11c4a2a3
            descr.ledalo _ FALSE ;                                  11c4a2a4
          END ;                                                     11c4a2b
    = lstrin:                                                       11c5
      BEGIN                                                         11c5a
        mxchar _ [value].M ;                                        11c5a1
        stradr _ getstring ( mxchar, zone ) ;                       11c5a2
        *stradr* _ *value* ;                                        11c5a3
        descr.ledval _ stradr ;                                     11c5a4
        descr.ledtyp _ lstrin ;                                     11c5a5
        descr.ledimd _ FALSE ;                                      11c5a6
        descr.ledalo _ TRUE ;                                       11c5a7
      END ;                                                         11c5b
    = llist:                                                        11c6
      BEGIN                                                         11c6a
        size _ [value].M ;                                          11c6a1
        blkadr _ getblk ( size, zone ) ;                            11c6a2
        blkadr[bhl] _ [value] ;                                     11c6a3
        FOR i _ 1 UP UNTIL > [value].L  DO                          11c6a4
```

list runtime program code

```
            blkadr[bhl+lhl+i] _ makelm ( [value][i],ledtyp,
            [value][i],ledval ) ;                                    11c6a4a
         descr.ledval _ &blkadr + bhl + lhl                          11c6a5
         descr.ledtyp _ llist ;                                      11c6a6
         descr.ledimd _ FALSE ;                                      11c6a7
         descr.ledalo _ TRUE ;                                       11c6a8
      END ;                                                          11c6b
   = lblock:                                                         11c7
      BEGIN                                                          11c7a
         size _ [value],blklength ;                                  11c7a1
         blkadr _ getblk ( size, zone ) ;                            11c7a2
         FOR i _ 0 UP UNTIL > size DO                                11c7a3
            blkadr[i+bhl] _ value[i+bhl] ;                           11c7a3a
         descr.ledval _ &blkadr ;                                    11c7a4
         descr.ledtyp _ lblock ;                                     11c7a5
         descr.ledimd _ FALSE ;                                      11c7a6
         descr.ledalo _ TRUE ;                                       11c7a7
      END ;                                                          11c7b
   ENDCASE ;                                                         11d
return ( descr ) ;                                                   12
(freelm) PROCEDURE ( list REF, index ) ;                             13
   % argument / result types %                                       13a
      % list = ADDRESS %                                            13a1
      % index = INTEGER %                                           13a2
   % frees any allocated storage associated with this element and
   sets the descriptor for this element to the null descriptor %     13b
```

list runtime program code

```
   IF list[index].ledalo THEN                                          13c
      BEGIN                                                            13c1
         CASE type OF                                                  13c1a
            = lnewde:                                                  13c1a1
               err ( s"list runtime package -  freelm : newdescriptor
               " ) ;                                                   13c1a1a
            = lnull:                                                   13c1a2
               err ( s"list runtime package - freelm : null " ) ;      13c1a2a
            = ldescr:                                                  13c1a3
               err ( s"list runtime package - freelm : descriptor " )
               ;                                                       13c1a3a
            = linteg:                                                  13c1a4
               BEGIN                                                   13c1a4a
                  blkadr _ list[index].ledval - bhl ;                  13c1a4a1
                  freeblk ( blkadr, zone ) ;                           13c1a4a2
               END ;                                                   13c1a4b
            = lstrin:                                                  13c1a5
               BEGIN                                                   13c1a5a
                  stradr _ list[index].ledval ;                        13c1a5a1
                  freestring ( stradr, zone ) ;                        13c1a5a2
               END ;                                                   13c1a5b
            = llist:                                                   13c1a6
               BEGIN                                                   13c1a6a
                  listad _ list[index].ledval ;                        13c1a6a1
                  nulist ( listad, FALSE, 0, 0 ) ;                     13c1a6a2
               END ;                                                   13c1a6b
```

list runtime program code

```
        = lblock:                                                    13c1a7
            BEGIN                                                    13c1a7a
               blkadr = list[index],ledval ;                         13c1a7a1
               freeblk ( blkadr, zone ) ;                            13c1a7a2
            END ;                                                    13c1a7b
      END ;                                                          13c2
   list[index] = nulldescr ;                                         13d
   return ;                                                          13e
(ckdesc) PROCEDURE ( descr ) ;                                       14
   % argument / result types %                                       14a
      % descr = WORD %                                              14a1
   %ckdesc checks a list element descriptor to see that it meets some
   criteria, for example that the type is in range and the flag bits
   are reasonable. ckdesc returns true if all is well, false
   otherwise. %                                                      14b
   IF descr,ledtyp < ldescr OR descr,ledtyp > lblock THEN return (
   FALSE ) ;                                                         14c
   IF descr,ledimd AND descr,ledalo THEN return (FALSE) ;            14d
   %one could check address range depending on local or global
   declared vs allocated storage. %                                  14e
   % one could ckeck to see that no unused bits are on. %           14f
   return ;                                                          14g
```

list runtime program code

(J25815) 30-APR-75 23:34;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, LIST-RUNTIME.NLS;11, >, 28-APR-75 13:56 JBP ;;;;####;

NSW Files -- Package, Format, Types, Movement

The File Package 1

Introduction 1a

This definition of the file package stems from a reconsideration of the needs and constraints of the NSW implementation. The main protion of these conclusions were reached at a meeting in late March. 1a1

The file package was examined and the essential features abstracted. This resulted in a small set of procedures, and the elimination of the access control aspects of the earlier specification. Also the abliity to access portions of files and route the file data on various paths was eliminated. 1a2

The relation between file package directories and Tenex directories is one to one. The access rights to directories and files that a caller on a file package has are exactly those of the user-password-account that the process containing the file package is logged in with. 1a3

The file package is an interface to the regular operating system file system and uses its access controls. 1a4

That is This definition to say that there is no attempt to build up a virtual file system at the file package level. 1a5

Definitions: 1b

The following arguments are used in the subsequent prodcedure definitions: 1b1

name - the complete name of a file in host dependent syntax 1b2

name - CHARSTR

In Tenex this includes the version number.

class - a partially specified file name that indicates a set of files in host dependent syntax. 1b3

class - CHARSTR

This is really only a special type of "name" as defined above.

In Tenex this is the star (*) notation.

directory - the name of a directory (or directory hierarchy) in host dependent syntax 1b4

directory - CHARSTR

Directory = EMPTY should default to the login directory.

Note that in Tenex the directory is not enclosed in angle brackets <>. The WM has to be able to use the same string for a login argument.

password - the secret word that gives access. 1b5

password - CHARSTR

workspace - a directory password pair 1b6

workspace - LIST (directory, password)

filename - the fully qualified name of a file 1b7

filename - LIST (workspace, name)

classname - the fully qualified name of a class of files 1b8

classname - LIST (workspace, class)

filelist - a list of file names 1b9

filelist - LIST (filename, ...)

classlist - a list of file classes 1b10

classlist - LIST (classname, ...)

srclist - list of source files 1b11

srclist - filelist

disp - the disposition of the source files, either DELETE or RETAIN. 1b12

disp - BOOLEAN [DELETE = FALSE / RETAIN = TRUE]

Note that DELETE makes the operation a rename, while RETAIN makes the operation a copy.

chnl - a port handle. 1b13

chnl - INDEX

If chnl is an argument of a procedure the data generated by (or received by) the procedure is transmitted on that physical channel .

filetypelist - a list of file types associated with the filelist that indicate the physical type of the file and the format of the pcp encoded transmission of the file. The type is represented by a small integer. 1b14

filetypelist - LIST (INDEX, ...)

These file types must be enumerated soon.

Procedures: 1c

Listdir (classlist, EMPTY -> filelist) 1c1

The names of the set of files indicated by CLASSLIST are returned in the result FILELIST.

This routine would accept (workspace, name) pairs of the following variety: (workspace, fileclass) which would do the TENEX star thing for that workspace, (workspace, name) which really asks if that file exists, (workspace, empty) which lists all files in that workspace, (empty, empty) which lists all files in the connected workspace, (empty, fileclass) which does the star thing for the connected workspace, etc. etc.

Listdir (classlist, chnl -> EMPTY) 1c2

The names of the set of files indicated by CLASSLIST are transmitted via the physical channel indicated by CHNL.

Deletefiles (filelist -> EMPTY) 1c3

The files specified in FILELIST are deleted.

Deletefiles (classlist -> filelist) 1c4

The files specified in CLASSLIST are deleted, the names of the deleted files are reported in in FILELIST.

Some interesting classes are: (workspace, *,*) might clear the entire workspace, as might (workspace, empty).

Localxfer (srclist, disp, classlist -> filelist) 1c5

The files specified by SRCLIST are assigned names and stored as indicated in CLASSLIST. When a name in CLASSLIST is incomplete a new unique name is generated to complete the name. The actual names used to store the files are returned in the FILELIST result.

The retention or deletion of the source files is indicated by DISP. All files in SRCLIST have the same DISP.

Localxfer (srclist, disp, filelist => EMPTY) 1c6

The files specified by SRCLIST are stored as indicated by FILELIST.

The retention or deletion of the source files is indicated by DISP. All files in SRCLIST have the same DISP.

Getfiles (srclist, filetypelist, disp, chnl) 1c7

The files are sent on the physical channel indicated by CHNL as specified by SRCLIST.

The type information in FILETYPELIST is used to determine the mapping from storage format to transmission format for the files.

The retention or deletion of the source files is indicated by DISP. All files in SRCLIST have the same DISP.

Putfile (filelist, filetypelist, chnl => EMPTY) 1c8

The files received on the physical channel indicated by CHNL are assigned the names and entered into workspaces as indicated by FILELIST.

The type information in FILETYPELIST is used to determine the storage format for the files.

Putfile (classlist, filetypelist, chnl => filelist) 1c9

The files received on the physical channel indicated by CHNL are assigned names as indicated by CLASSLIST. When an entry in CLASSLIST is not complete a unique name is assigned to complete the name. The list of new file names is reported in the FILELIST result.

The type information in FILETYPELIST is used to determine the storage format for the file.

Discussion: 1d

A convention to be followed whenever two parallel lists are supplied as arguments is that if the second list runs out before the first list, then the last element of the second list is to repeated for every remaining element of the first list. 1d1

Whenever an input argument specifies a class of files or incompletely specifies a file name, then the procedure is to return the complete list of actual file names. Only when the input argument completely specifies all file names completely does the procedure return the EMPTY result. 1d2

Another convention is that the procedures of the file package are to make help returns to their caller on any error. 1d3

Examples of errors tha could be so reported:

Source file does not exist

Access control prevent your use of that file

Unrecoverable I/O error

The intent of the help return is to have the file package procedures report the failure of an operation on a per file basis, that is, the help return can indicate the specific file in error. This then allows the caller to resume or abort the procedure with full knowledge of how far it got, or which files were not processed.

We can identify three possibilities after error detection and a help call:

1) skip that element and proceed on to next one,

2) abort the whole call, with or without trying to undo what you've already done,

and 3) try same element again with newly specified parameters.

The exact nature of these help calls will become clearer as the implementations proceed.

Transmission Format 2

Files are transmitted between file packages in the format of PCP data structures. Each file is transmitted as a list of two structures: a file descriptor block, and the actual file datastructure. 2a

transfile = LIST (filedesc, filedata) 2a1

The file descriptor block is a list of the filetype, the actual file length in bits, the number of records in the file, and the maximum length of a record. 2b

filedescr = LIST (filetype, bitlen, numrec, maxrec) 2b1

filetype = INDEX

bitlen = INTEGER

numrec = INTEGER

maxrec = INTEGER

If the actual number of bits in the file is unknown then this field is set to zero. 2c

File Types 3

This section specifies the currently defined physical file types within the NSW, and specifies the PCP encodings used to communicate the files among various PCP processes. The actual PCP format , i.e. PCPB36 , PCPB8 or PCPTXT, used on the connection must be agreed upon between the PCP IPC modules at the two ends of a physical channel but is irrelevent to this discussion. 3a

Physical File Attributes: 3b

The Physical file type is specified by three attributes: 3b1

DATA TYPE:

This attribute has the value CHARACTER or BINARY and specifies whether the file is comprised of character strings or bit strings. Since it is clearly possible to encode any file as either data type this attribute is not an absolute constraint on the contents of the file but rather an indication of the most advantageous encoding to use.

RECORD TYPE:

Record type indicates the record structure of the file in the originating process. It has the following legal values.

FIXED: The file consists of records of fixed size.

VARIABLE: The file consists of records of variable size.

STRUCTURE TYPE:

This attribute specifies whether the file is a simple sequence of records or whether there is a more complex record structure. The legal values are:

SEQUENTIAL: The file is transmitted as a sequence of records.

SPARSE: Each record carries a record number along with it. The list of pairs (Record number,record data) are simply ordered on record number. That is the record number of each record is greater than that of its predecessor. The record number of the first record cannot be less than zero.

RANDOM: Each record carries a record number along with it. The constraints on record numbers are that they are unique, and that record numbers are non-negative.

PCP encodings of files 3c

CHARACTER (FIXED / VARIABLE) SEQUENTIAL 3c1

LIST (%datarecord% CHARSTR, ...)

CHARACTER (FIXED / VARIABLE) (SPARSE / RANDOM) 3c2

LIST (LIST (%recordnumber% INTEGER, %datarecord% CHARSTR), ...)

BINARY (FIXED / VARIABLE) SEQUENTIAL 3c3

LIST (%datarecord% BITSTR, ...)

BINARY (FIXED / VARIABLE) (SPARSE / RANDOM) 3c4

LIST (LIST (%recordnumber% INTEGER, %datarecord% BITSTR), ...)

Use Types: 3d

In addition to Physical File Type each NSW file also has an attribute called use type which is assigned at creation time by the creating tool. This attribute is used to give an indication of the semantic content of the file. It is our intention that the WM will store a matrix whose entries are of the form (process name, package name, procedure name) and that is indexed by (source file physical type, source file use type, destination file physical file type, destination file use type). The procedure thus indexed has as parameters (source file name, destination file name) and will either return TRUE indicating the destination file has been successfully created and entered into the NSW file system or FALSE indicating failure of the conversion procedure. 3d1

In the initial phases of the NSW it is expected that this conversion matrix will be extremely sparse. Indeed many of the elements of this matrix will never be implemented, for example the task of converting a 360 cobol object file into a fortran source file seems well beyond the initial design goals. However some of the entries in this conversion matrix will be supplied by the utility packages of each TBH (NSW Tool Bearing Host). In addition tool purveyers may find it in their interest to supply elements of the matrix corresponding to the use types most commonly created or requested by their tool. This allows a potential user to integrate the use of this tool more easily with other tools he uses. 3d2

In the case where the use type of a file is undefined or where the element of the conversion matrix needed is empty it seems advantages to supply default conversions based soely on physical file type. In fact the set of conversions based solely upon physical file type should form the minimum set of conversions provided by the file package of each TBH. 3d3

The following is a first cut at defining the conversions based solely on physical file type. 3d4

Physical File Type conversions: 3d5

The following conversions are defined separately for each physical file attribute. Conversion between physical file types is accomplished by performing each of the three possible translations (one for each attribute) concurrently.

Some of the following conversions take arguements which specify conversion parameters, I am as yet unclear exactly who specifies these , how and when, It seems that the requestor and supplier of the file must negotiate the proper values for these parameters, In the case where the requestor is a user this is fairly straight forward, however the case in which the requestor is a tool which in turn might want to consult the user is less clear,

Attribute conversion primitives:

CHARACTER => BINARY

Each Character is simply converted to an eight bit byte containing the ASCII character code in the low order 7 bits,

BINARY => CHARACTER

Treat each 8 bit bite as containing one ASCII character in the low order 7 bits,

FIXED => VARIABLE

preceed each record by the character/bit count for the record,

VARIABLE => FIXED

PARAMETERS (fixedrecordlength %INTEGER%, fillcharacter %CHARSTR%, break %BOOLEAN%, append %BOOLEAN)

If the input record is shorter than the requested fixedrecordlength and append is FALSE the record is padded with the fill character/bit , If however append is TRUE a new input record is fetched and is inserted in the current output record beginning with the next unused character/bit position, This continues until the current output record is full at which point a new record is begun if break is true, otherwise the unused portion of the input buffer is discarded,

If the input record is longer than the fixedrecordlength and break is FALSE the record is truncated with the truncated portion being lost, If However break is TRUE a next record is begun, This is repeated until the entire input record has been processed, If append is FALSE then the last fixed record is padded, otherwise the next input record continues filling this current fixed length record and this process is continued until the last input record is processed at which point the last output record is padded if necessary.

SEQUENTIAL => SPARSE and SEQUENTIAL => RANDOM

Parameters (initrecnum, recinc)

Each input record is assigned a record number beginning with initrecnum and incrementing by recinc,

SPARSE => SEQUENTIAL

Record numbers are simply discarded,

RANDOM => SEQUENTIAL

The receiving process collects all the input records, sorts them by record number if necessary and then discards the record numbers,

SPARSE => RANDOM

no conversion needed SPARSE is a proper subset of RANDOM

RANDOM => SPARSE

The receiving process collects all the input records and sorts them by record number if necessary,

Declarative Specifications 3e

The file type is encoded into a small integer specified using the PCP data type INDEX when passed as an argument, 3e1

File Types 3e2

Type 1

CHARACTER FIXED SEQUENTIAL

Type 2

CHARACTER VARIABLE SEQUENTIAL

Type 3

CHARACTER FIXED SPARSE

Type 4

CHARACTER FIXED RANDOM

Type 5

CHARACTER VARIABLE SPARSE

Type 6

CHARACTER VARIABLE RANDOM

Type 7

BINARY FIXED SEQUENTIAL

Type 8

BINARY VARIABLE SEQUENTIAL

Type 9

BINARY FIXED SPARSE

Type 10

BINARY FIXED RANDOM

Type 11

BINARY VARIABLE SPARSE

Type 12

BINARY VARIABLE RANDOM

Type 13

Card Image

CHARACTER FIXED SEQUENTIAL

where each CHARSTR is of length 80 (and does not include <CR> <LF> to signal end of card). In this type of file there are no format effectors at all.

Type 14

Text Line

CHARACTER VARIABLE SEQUENTIAL

where each CHARSTR ends with the character pair <CR> <LF>, all the ASCII format effectors [<FF>, <CR>, <LF>, <HT>, <VT>, <BS>] are allowed (See document format 2, RFC 678 -- 31524,).

Type 15

Tenex Page

BINARY VARIABLE SPARSE

where each BITSTR is a maximum of 18432 (512*36) bits, and missing records are 18432 bit chunks.

Type 16

Print Line

CHARACTER VARIABLE SEQUENTIAL

where each CHARSTR ends with the character pair <CR> <LF>, and printer format is directed by the ASCII format effectors <FF>, <CR>, and <LF> (See document format 3, RFC 678 -- 31524,).

Type 17

Print ASA

CHARACTER VARIABLE SEQUENTIAL

where each CHARSTR contains as its first character an ASA carriage control character, the remaing characters in the CHARSTR consist only of the printable characters and blank.

Type 18

Print Noformat

CHARACTER VARIABLE SEQUENTIAL

where each CHARSTR consist only of the printable characters and blank.

Copying Files 4

Introduction 4a

This is a description of how files are copied (or moved) both inside the NSW and across the NSW boundary. 4a1

Copying Files Within the NSW 4b

Introduction 4b1

This is a description of my model of the procedures involved in moving a NSW file from one file package controlled location to another file package controlled location.

This package is designed to support the implementation of the Works Manager primitive COPY.

The Model 4b2

COPY (srchost, srcfilelist, srctypelist, disp, dsthost, dstfilelist, dsttypelist)

A file is specified by three character strings: the directory, the password, and the name.

filename = LIST (workspace, name)

workspace = LIST (directory, password)

File names may be collected together in lists and the list passed as an argument if the same function is to be applied to that set of files.

filelist = LIST (filename, ...)

This routine looks up the two host references and determines the source and destination processes.

ph1 = source process

ph2 - destination process

This routine creates a channel between the source and destination file packages (which are already open) by calling on the local (to the this process) process management package (PMP).

CRTPHYCHN (ph1, ph2 => poh1, poh2, pcn)

poh1 - handle by which ph1 knows the channel

poh2 - handle by which ph2 knows the channel

pcn - handle by which the this routine knows the channel

This routine calls the GETFILES procedure in the file package at the source location.

disp - RETAIN for copy, or DELETE for move

Getfiles (srcfilelist, srctypelist, disp, poh1)

The files are sent on the physical channel indicated by poh1 as specified by srcfilelist.

The type information in srctypelist is used to determine the mapping from storage format to transmission format for the files.

The retention or deletion of the source files is indicated by disp. All files in srcfilelist have the same disp.

The file access parameters are checked.

Getfiles actually reads the files from the local file system and send the files in the PCP format indicated by srctypelist via the IPC procedure SNDMSG, generally this will require a series of file reads and sndmsgs.

message - a portion of a file

SNDMSG (poh1, message)

This routine calls the Putfiles procedure in the file package at the destination location.

Putfiles (dstfilelist, dsttypelist, poh2 => EMPTY)

The files received on the physical channel indicated by poh2 are assigned the names and entered into directories as indicated by dstfilelist.

The type information in dsttypelist is used to determine the storage format for the files.

The file access parameters are checked.

Putfiles actually receives the files via the IPC procedure RCVMSG and stores the files to the local file system, generally this will require a series of rcvmsgs and file stores.

message = a portion of a file

RCVMSG (poh2, message)

Comments 4b3

The procedures Getfiles and Putfiles must be implemented such that there is careful consideration of the asynchronous timing of the calls.

If parallel calls are made to a pair of file packages the caller must be careful to provide a distinct channel for each simultaneous transfer requested.

Copying Files Across the NSW Boundary 4c

Introduction 4c1

This describes the package that is called on by the Works Manager to move files between an NSW file location and an arbitrary ARPANET file location (or between two arbitrary ARPANET file locations). This package presents a DPS (PCP) interface to its callers and utilizes the FPTFRK program (23649,).

This package is designed to support the implementation of the Works Manager primitives EXPORT, IMPORT, and TRANSPORT.

The Model 4c2

COPY (srchost, srclog, srcfilelist, dsthost, dstlog, dstfilelist)

where srclog = LIST (srclname, srclpass, srclacct)

and dstlog = LIST (dstlname, dstlpass, dstlacct)

and srcfilelist = LIST (srcfile, ...)

where srcfile = the source file's pathname in host dependent syntax

and dstfilelist = LIST (dstfile, ...)

where dstfile = the destination file's pathname in host dependent syntax

This routine calls on its inferior FTPFRK to transfer the files.

Begin ()

Open (srchost)

Login (srclname, srclpass, srclacct)

Shelv (srcstate)

Open (dsthost)

Login (dstlname, dstlpass, dstlacct)

Lbl (dststate)

Ncpy (srcstate, srcfile, dststate, dstfile, xfermode)

where xfermode = APPROPRIATE

This step may be repeated until the srcfilelist is exhausted.

Close ()

Mount (srcstate)

Close ()

End ()

Comments 4c3

If the source file is in a file package controlled location and the destination file is not this is the Works manager Export action.

If the source file is not in a file package controlled location and the destination file is this is the Works manager Import action.

If neither the source file or the destination file are in file package controlled locations this is the Works Manager Transport action.

If the source file and destination file are both in file package controlled locations this transfer will still work but the knowledge of types will be lost and type conversions will not occur except for the limited conversions embodied in ARPANET File Transfer Protocol.

It will be generally useful to provide a Procedure Call interface to all of the routines of the FTPFRK.

JBP 1-MAY-75 03:36 25816

NSW Files -- Package, Format, TYpes, Movement

(J25816) 1-MAY-75 03:36;;; Title: Author(s): Jonathan B. Postel/JBP;
Distribution: /JBP([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk:
JBP; Origin: < POSTEL, NSW-FILES,NLS;2, >, 1-MAY-75 03:28 JBP
;;;;####;

JAKE 1-MAY-75 17:09 25817

NIC Reporting 74-75

Dear Stoney, Have checked into the particulars on the 1974-75 NIC contract and as far as I know CFSRs are not required on that contract. However, Spencer Floyd of our contracts office has sent you a brief summary of the financial status of the contract by mail. With regard to the QMRs - the contract was signned Feb. 6, 1975 and the first QMR is due 100 days after signing which would make the due date for the first report approximately May 19, 1975. That of course is the letter of the law but not necessarily the spirit. I will try to get a report out sometime next week to give you an idea where things stand. If you need other bits and pieces please let me know. Regards, Jake 1

NIC Reporting 74-75

(J25817) 1-MAY-75 17:09;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /DLS([INFO-ONLY]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) DVN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

BEV POOH 1-MAY-75 18:12 25818

Proposed Preface Organization

we are beginning to revise the preface so let us know if something doesn't sound right.

Proposed Preface Organization

Basic approach of NLS Preface: We are designing the preface as a descriptive document for the brand-new user to pick up before he ever tries to operate a terminal. We want to limit the information in it to the absolute essentials only. Within the Preface will be numerous references to scenarios, which the new user will use to begin working on the terminal. 1

The scenarios will be in a format somewhere between that of the Primer and the scenarios developed by Susan. They will all be referenced in the Preface, and will be modular (i.e. each one integral, and none overlapping). The scenarios will be designed as teaching guides, with only the basic information to perform simple tasks included. The idea is that the new user can take them to his terminal and work through the operations, as well as use them for quick and easy reference on basic procedures later on. Examples: logging in; editing commands; going from system to system; sendmail. 1a

We will rewrite the present Preface to simplify it. This will not entail too much work, as we will use the more pictorial forms of conveying information which Ann has already developed (e.g. instead of long description of structural elements, a diagram showing the major components, and how they fit together). 1b

The major sections to be covered in the Preface are as follows (those to have scenarios linked to them are shown): 2

Intro. to NLS (scenario on logging in) 2a

Structure and file organization 2b

Commands (and command syntax) 2c

Help (scenario) 2d

Editing (probably 2 or more scenarios building up in degree of complexity) 2e

TNLS addressing (probably more than one scenario: addressing; linking; getting around among files; using directories) 2f

Available Subsystems (tools) 2g

Readmail (scenario) 2g1

Sendmail (scenario) 2g2

Other Tools (list) 2g3

Proposed Preface Organization

Moving among tools, systems, etc, (e.g. how to negotiate between FEWM and NLS-8,9; NLS-8,9 and Readmail) (scenario) 2g4

Control characters (list) 2h

NOTES: 3

We would use the present version of the primer as one of the Editing scenarios, deleting the portion on Sendmail. 3a

The present section on addressing would be modified. The sections on 'addressing by structure' and 'frequently used address file' will be deleted. We should see if we can clarify some of this section by diagrams. 3b

*****INFORMATION WE NEED: 3c

Can we use the scenario Dirk/Kirk are going to write for the FEWM as our logging in scenario? 3c1

Do we need to write scenarios for both TNLS and DNLS? 3c2

Proposed Preface Organization

(J25818) 1-MAY-75 18:12;;; Title: Author(s): Beverly Boli, Ann Weinberg/BEV POOH; Distribution: /KIRK([ACTION]) DVN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: BEV; Origin: < BOLI, PREFACE/NLS,NLS;2, >, 30-APR-75 17:31 BEV ;;;;####;

JLE 2-MAY-75 13:00 25819

CLI - OSI Terminal Handler Specifications

Draft document used in discussions with ADR

Terminal File Manipulation Procedures 1

1a

Primary File Manipulation 1b

Introduction 1b1

At any point in time, a process has associated with it one and only one primary input file and one and only one primary output file. At initialization time, these will correspond to the physical terminal normally being used by a user. These are the files that are referenced when a process uses the file-ids PIFID and POFID. The following procedures allow processes to divert input or output to or from a physical terminal to other devices (e.g., a disk file to be read later or a command file for frequently performed functions). 1b1a

read-primary-file-handles (process-id => PIF-id, POF-id) 1b2

This procedure returns the file-ids of the current primary input and output files. If I/O has not been diverted, then these will be PIFID and POFID. 1b2a

FORMAT: 1b2b

process-id is INTEGER (see -- xxx.) 1b2b1

PIF-id is INTEGER 1b2b2

This will be the file-id for the current primary input file. 1b2b2a

POF-id is INTEGER 1b2b3

This will be the file-id for the current primary output file. 1b2b3a

write-primary-file-handles (process-id, PIF-id, POF-id) 1b3

This procedure changes the current primary input and output files. If a process does not wish to change its primary input file, it should use the file-id PIFID for the PIF-id parameter. It may perform a similar change for its primary output file. 1b3a

reset-primary-file-handles (process-id) 1b4

This procedure resets the primary input and output files for the specified process back to the initial values when the process started. 1b4a

write-terminal-file-characteristics (FILE*, file-characteristics-list) 1b5

This procedure enables a process to modify some of the physical characteristics of specified file. 1b5a

FORMAT: 1b5b

file-characteristics-list is LIST( ) 1b5b1

The value of this list will be specified later; however, it will probably include such things as tab stops, etc. 1b5b1a

read-terminal-file-characteristics (FILE* => file-characteristics-list) 1b6

This procedure enables a process to determine the logical class and the physical characteristics of the terminal to which file-id refers. 1b6a

FORMAT: 1b6b

file-characteristics-list is LIST( ) 1b6b1

The value of this list will be specified later; however, it will probably include such things as the logical terminal class, whether or not the terminal has lower case, etc. 1b6b1a

reset-process-terminal-file (process-id) 1b7

This procedure will reset the PIF and POF files for the specified process back to their initial state (see above -- xxx). 1b7a

FORMAT: 1b7b

process-id is INTEGER 1b7b1

1b8

Input Text Manipulation 1c

input-byte (FILE* => byte-value) 1c1

This procedure will input a character from the specified file. 1c1a

FORMAT: 1c1b

byte-value is CHARSTR 1c1b1

This result is the value of the input byte. 1c1b1a

output-byte (FILE*, byte-value) 1c2

This procedure will output a character to the specified file. When the specified file is both the primary file and a terminal file, characters written by this procedure will appear in the window designated to receive teletype output. 1c2a

FORMAT: 1c2b

byte-value is CHARSTR 1c2b1

This parameter specifies the value of the output byte. 1c2b1a

input-string (FILE*, termination-condition -> string-value) 1c3

This procedure will input a number of successive characters from the specified file. 1c3a

FORMAT: 1c3b

termination-condition is LIST(%count% INTEGER, %chars% STRING) 1c3b1

This parameter causes the input procedure to terminate transmission under the following conditions: 1c3b1a

if count is zero, characters are input until one matches any byte in the chars string. 1c3b1a1

if chars is a null string, "count" characters are input. 1c3b1a2

if both count and chars are specified, characters are processed until either "count" characters have been input or one matches any byte in the chars string. 1c3b1a3

string-value is CHARSTR 1c3b2

This result is the value of the input string. 1c3b2a

output-string (FILE#, termination-condition, string-value) 1c4

This procedure will write a number of successive characters to the specified file. When the specified file is both the primary file and a terminal file, characters written by this procedure will appear in the window designated to receive teletype output. 1c4a

FORMAT: 1c4b

termination-condition is LIST(%count% INTEGER, %chars% STRING) 1c4b1

This parameter causes the output procedure to terminate transmission under the following conditions: 1c4b1a

if count is zero, characters are output until one matches any byte in the chars string. 1c4b1a1

if chars is a null string, "count" characters are output. 1c4b1a2

if both count and chars are specified, characters are processed until either "count" characters have been output or one matches any byte in the chars string. 1c4b1a3

string-value is CHARSTR 1c4b2

This parameter is the value of the output string. 1c4b2a

1c5

Pseudo Interrupts 1d

Introduction 1d1

Just as hardware interrupts permit the efficient utilization of a computer shared by numerous processes, another form of interrupt is useful to users working at terminals. This second form of interrupt is called "pseudo-interrupt" so that it is not confused with the first. A "pseudo-interrupt" occurs when the user types one of a set of pseudo-interrupt characters at the terminal. By software interpretation, the priority of this pseudo-interrupt character is examined to determine whether to immediately suspend the process currently running or to postpone the pseudo-interrupt until this higher priority process has concluded. (In the case of class 0 terminals (half-duplex,

line at a time), the pseudo-interrupt character is not detected until the user has transmitted the entire line.) After the pseudo-interrupt has been processed by the appropriate pseudo-interrupt procedure, control resumes with the originally suspended process. 1d1a

At a later time, consideration should be given to having pseudo-interrupts associated with non-terminal files. In the initial NSW implementation, however, an error will be generated when the specified file refers to a non-terminal file. 1d1b

Enable-Pseudo-Interrupts (FILE*) 1d2

This procedure "enables" (i.e., turns on) the pseudo-interrupt system for the specified file. Individual pseudo-interrupt characters can be activated and deactivated independently from the status of the pseudo-interrupt system for the specified file. However, until the PSI system is enabled for a file, none of the activated characters input from that file will generate pseudo-interrupts and in fact will be processed as normal input. 1d2a

Disable-Pseudo-Interrupts (FILE*) 1d3

This procedure "disables" (i.e., turns off) the pseudo-interrupt system for the specified file. 1d3a

Activate-char-as-PSI (FILE*, char, priority, proc-name =>
psi-char-id) 1d4

This procedure specifies that a pseudo-interrupt should be generated when the PSI system is enabled and the specified character is read from the specified file. It also defines the priority of the pseudo-interrupt to be associated with that character and the procedure to be called when the PSI is generated. 1d4a

FORMAT: 1d4b

char is CHARSTR 1d4b1

priority is INTEGER [P0=0 / P1=1 / P2=2 / P3=3] 1d4b2

This parameter specifies the priority to be associated with the PSI for this character. If a pseudo-interrupt of priority j is in progress, then only PSIs of higher priority k will be initiated. Any

PSIs of equal or lower priority will be remembered for processing after the current PSI is "debreaked". 1d4b2a

proc-name is INTEGER 1d4b3

This parameter specifies the address of a local procedure that should be called when the PSI is generated. It is important to note that this assumes knowledge of the language in which the called procedure is written so that the proper type of procedure call activation can be made. Initially in the NSW, we will assume this language to be LIO. At a later time, this may have to be replaced with an address to receive direct control rather than a procedure to be called. 1d4b3a

psi-char-id is INTEGER 1d4b4

This result is an id that is to be used for future references to this PSI and is local to the specified file. 1d4b4a

Deactivate-char-as-PSI (FILE*, Psi-char-id) 1d5

This procedure deactivates the PSI-processing procedure associated with the specified psi-char-id. After deactivation has been performed, the appearance of this character in the FILE* input stream will not cause a pseudo-interrupt. 1d5a

FORMAT: 1d5b

psi-char-id is INTEGER (see above) 1d5b1

Read-char-psi-status (FILE*, psi-char-id -> char, priority, proc-name) 1d6

This procedure returns the current status of the specified file PSI system associated with the specified psi-char-id. 1d6a

FORMAT: 1d6b

psi-char-id is INTEGER 1d6b1

char is CHARSTR 1d6b2

This result is the 8-bit character associated with the specified psi-char-id. 1d6b2a

priority is INTEGER 1d6b3

This result is the priority associated with the specified psi-char-id. 1d6b3a

proc-name is INTEGER 1d6b4

This result is the address of the local PSI-processing procedure associated with the specified pdi-char-id. 1d6b4a

Read-file-psi-status (FILE* => file-psi-status) 1d7

This procedure returns the current status of the PSI system for the specified file (i.e., is it enabled or disabled) and a list of all activated character psi-char-ids and their associated PSI-processing procedures. 1d7a

FORMAT: 1d7b

file-psi-status is LIST(psi-status, psi-char-list) 1d7b1

psi-status is BOOLEAN [ON=TRUE / OFF=FALSE] 1d7b2

psi-char-list is LIST(LIST(psi-char-id, proc-name), ...) 1d7b3

1d8

Echo control 1e

Introduction 1e1

At initialization time, the first 128 character codes will be echoed directly without any transformations applied and none of the second 128 codes will be echoed. The following two procedures are provided to modify this initial state. 1e1a

write-echo-status-file (FILE*, echo-status) 1e2

This procedure specifies how characters should be echoed when typed at a terminal. (Note that specifying anything but no echoing for class 0 (line at a time) terminals can lead to unwanted results appearing on the terminal. Moreover, only physical terminals (as opposed to terminal disk files) should have echoing performed.) 1e2a

FORMAT: 1e2b

echo-status is LIST(LIST(class / char, echo-str), ...
) 1e2b1

class is INTEGER [CONTROL=0 / ALPHA=1 / NUM=2 / PUNC=3 /
FIRST-128=4 / SECOND-128=5 / SPACE=6 / ALL=7] 1e2b2

char is CHARSTR 1e2b3

This parameter is the 8-bit character which should be replaced by the defined echo string. 1e2b3a

echo-str is CHARSTR 1e2b4

This parameter is the string that should be echoed when the specified character is input. This can be a null string indicating that the original character is not echoed. It would be very useful to have a meta language to do things such as echoing a class of characters with a mapping of those characters, (e.g., echo control-L as "<^L>"). 1e2b4a

read-echo-status-file (FILE* => echo-status) 1e3

This procedure returns the current echo status for the specified file. 1e3a

FORMAT: 1e3b

echo-status is (see above) 1e3b1

1e4

Character Translation Control 1f

Introduction 1f1

All 256 character codes normally are given to a process exactly as input from the current input stream. The following two procedures are provided to modify this initial state. 1f1a

write-input-char-trans (FILE*, input-trans-status) 1f2

This procedure defines specific character translations that should be performed for the specified file. Characters coming from the FILE* input stream will be translated according to a table containing all translation information before being given to the requesting procedure. 1f2a

FORMAT: 1f2b

input-trans-status is (see echo-status above) 1f2b1

read-input-char-trans (FILE* -> input-trans-status) 1f3

This procedure returns the current status of the input character translation table for the specified file. 1f3a

FORMAT: 1f3b

input-trans-status is (see echo-status above) 1f3b1

1f4

Window Manipulation 1g

Introduction 1g1

Windows can be manipulated on display terminals only (classes 3, 4,and 5). An attempt to manipulate windows on class 0, 1, or 2 (non-display) terminals or on non-terminal files will generate an error. 1g1a

allocate-window (FILE*, window-parms -> window-id) 1g2

This procedure allocates a window with the specified characteristics in the specified file. 1g2a

FORMAT: 1g2b

window-parms is LIST(type, bounds, priority, visibility, hit-sensitivity, typewriter) 1g2b1

type is INTEGER [SEQUENTIAL=0 / RANDOM=1 / CURSOR=2] 1g2b2

bounds is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1g2b3

This list specifies the virtual coordinates of the lower left and upper right hand corners of the window. 1g2b3a

priority is INTEGER [FLOAT=-1 / P0=0 / P1=1 / P2=2 / P3=3] 1g2b4

This parmeter specifies the priority with which the specified window is displayed. The allocation of a window with higher priorty which shares virtual coordinate space with window(s) of lower priority

causes these others to become temporarily invisible. The FLOAT priority indicates that the specified window has equal visibility priority with any windows that overlap (e.g., the default cursor window). In this case, the contents of overlapped windows are superimposed (if possible) on the terminal. 1g2b4a

visibility is BOOLEAN [VISIBLE=TRUE / INVISIBLE=FALSE] 1g2b5

This parameter specifies whether or not the contents of the window are to be visible to the viewer. 1g2b5a

hit-sensitivity is BOOLEAN [SENSITIVE=TRUE / INSENSITIVE=FALSE] 1g2b6

This parameter specifies whether or not the atoms that make up this window are "hit sensitive" to the select primitive described elsewhere. 1g2b6a

typewriter is BOOLEAN / empty 1g2b7

If this parameter is true (which is valid for sequential windows only), then any characters output to the file POFID which are not part of any of the other commands specified below (e.g. part of a write-string command) are sent to this window. 1g2b7a

window-id is INTEGER 1g2b8

This result is the id that should be used for future reference to this window and is local to the the specified file. 1g2b8a

deallocate-window (WINDOW*) 1g3

This procedure deallocates the specified window, and any atoms that belong to the window. 1g3a

FORMAT: 1g3b

If window-id is empty, this procedure deallocates all windows previously allocated by this process within the specified file. 1g3b1

manipulate-window (WINDOW*, manipulation-parms) 1g4

This procedure allows a process to manipulate a window. 1g4a

FORMAT: 1g4b

manipulation-parms is LIST(bounds, priority, visibility, hit-sensitivity, typewriter) 1g4b1

bounds is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1g4b2

This parameter can only be specified if the window being manipulated is a cursor window. If this is the case, then this list specifies the coordinates of the lower left and upper right hand corners for the window 1g4b2a

priority is INTEGER [FLOAT=-1 / P0=0 / P1=1 / P2=2 / P3=3] 1g4b3

visibility is BOOLEAN [VISIBLE=TRUE / INVISIBLE=FALSE] 1g4b4

hit-sensitivity is BOOLEAN [SENSITIVE=TRUE / INSENSITIVE=FALSE] 1g4b5

typewriter is BOOLEAN 1g4b6

read-window-parms (WINDOW* -> window-parms) 1g5

This procedure returns the status of the specified window in the specified file. 1g5a

FORMAT: 1g5b

window-parms is LIST() (see above) 1g5b1

write-error-window (FILE*, error-string) 1g6

This procedure writes a string into the error window of the specified file. 1g6a

FORMAT: 1g6b

error-string is CHARSTR 1g6b1

write-status-window (FILE*, status-string) 1g7

This procedure writes a string into the status window of the specified file. 1g7a

FORMAT: 1g7b

status-string is CHARSTR 1g7b1

1g8

Atom Manipulation 1h

allocate-atom (WINDOW*, atom-parms -> atom-id) 1h1

This procedure allocates an atom, with the specified characteristics, within the specified window. 1h1a

FORMAT: 1h1b

atom-parms is LIST(bounds, visibility, hit-sensitivity) 1h1b1

bounds is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1h1b2

This list specifies the virtual coordinates of the lower left and upper right hand corners of the atom. 1h1b2a

visibility is BOOLEAN [VISIBLE=TRUE / INVISIBLE=FALSE] 1h1b3

If the window in which this atom resides is currently INVISIBLE, then the atom will be invisible independent of the visibility parameter for the atom. On the other hand, if the owning window is VISIBLE, then the visibility of the atom is governed by this visibility parameter. 1h1b3a

hit-sensitivity is BOOLEAN [SENSITIVE=TRUE / INSENSITIVE=FALSE] 1h1b4

If the window in which this atom resides is currently INSENSITIVE, then the atom will be insensitive independent of the hit-sensitivity parameter for the atom. On the other hand, if the owning window is SENSITIVE, then the sensitivity of the atom is governed by this hit-sensitivity parameter. 1h1b4a

atom-id is INTEGER 1h1b5

This result is the id that should be used for all future references to the atom and is local to the specified window. 1h1b5a

deallocate-atom (ATOM*) 1h2

This procedure will deallocate the specified atom(s). 1h2a

FORMAT: 1h2b

If atom-id is empty, then this procedure will deallocate all the atoms in the specified window. 1h2b1

manipulate-atom-parms (ATOM*, new-atom-parms) 1h3

This procedure allows a process to modify several of the parameters associated with an atom. 1h3a

FORMAT: 1h3b

new-atom-parms is LIST(visibility, hit-sensitivity) 1h3b1

visibility is BOOLEAN [VISIBLE=TRUE / INVISIBLE=FALSE] 1h3b2

hit-sensitivity is BOOLEAN [SENSITIVE=TRUE / INSENSITIVE=FALSE] 1h3b3

read-atom-parms (ATOM* -> atom-parms) 1h4

This procedure returns the current parameters for the specified atom. 1h4a

FORMAT: 1h4b

atom-parms is LIST( ) 1h4b1

copy-atom (ATOM*-1, ATOM*-2) 1h5

This procedure copies the contents of atom-id-1 to atom-id-2. The entire contents of atom-id-2 are replaced by as much as will fit of atom-id-1. 1h5a

move-atom (ATOM*-1, ATOM*-2) 1h6

This procedure moves the contents of atom-id-1 to atom-id-2. The entire contents of atom-id-1 are deleted, and the entire contents of atom-id-2 are replaced by as much as will fit of atom-id-1. 1h6a

1h7

Segment Manipulation 1i

write-segment (ATOM*, segment-list -> segment-id-list) 1i1

This procedure writes the list of segments into the specified atom and returns a list of segment-ids. 1i1a

FORMAT: 1i1b

segment-list is LIST(segment, ...) 1i1b1

This parameter specifies the list of segments to be written in the specified atom. 1i1b1a

segment is LIST(acoors, segment-parms, segment-string) 1i1b2

This parameter specifies the individual characteristics of each segment. 1i1b2a

acoors is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1i1b3

This parameter specifies the positional coordinates of the segment relative to the atom origin. 1i1b3a

segment-parms is LIST(visibility, hit-sensitivity, highlight) 1i1b4

This parameter specifies the characteristics associated with each segment (e.g., visibility, hit-sensitivity, etc.). 1i1b4a

highlight is INTEGER [NORMAL=0 / HIGHLIGHT=1] 1i1b5

This parameter specifies whether or not the newly written string should be made to "stand-out". 1i1b5a

segment-string is CHARSTR 1i1b6

This parameter is the string to be written. 1i1b6a

segment-id-list is LIST(segment-id , ...) 1i1b7

This result list returns the appropriate segment-id for each individual parameter segment. 1i1b7a

read-segment (ATOM#, segment-id-list -> segment-list) 1i2

This procedure returns the contents of each segment specified in the segment-id-list. 1i2a

FORMAT: 1i2b

segment-id-list is LIST() (see above) 1i2b1

segment-list is LIST() (see above) 1i2b2

manipulate-segment (ATOM*, segment-id-list, segment-parms-list) 1i3

This procedure permits the modification of the parameters associated with each segment specified in the segment-id-list. 1i3a

FORMAT: 1i3b

segment-id-list is LIST(segment-id , ...) 1i3b1

This parameter list specifies the appropriate segment-id for each individual parameter segment. 1i3b1a

segment-parms-list is LIST(segment-parms) 1i3b2

This parameter list returns the parameters associated with each segment-id. 1i3b2a

1i4

Stream Manipulation 1j

write-stream (ATOM*, stream-list => stream-id-list) 1j1

This procedure writes the list of streams into the specified atom and returns a list of stream-ids. The information contained in these streams is not inspected in any way. Rather, it is transmitted in a "transparent" mode to the specified atom without any of the ancillary services typically performed by the OSI terinal handler. 1j1a

FORMAT: 1j1b

stream-list is LIST(stream, ...) 1j1b1

This parameter specifies the list of streams to be written in the specified atom. 1j1b1a

stream is LIST(acoors, stream-parms, stream-string) 1j1b2

This parameter specifies the individual characteristics of each stream. 1j1b2a

acoors is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1j1b3

This parameter specifies the positional coordinates of the stream relative to the atom origin. 1j1b3a

stream-parms is LIST(visibility, hit-sensitivity) 1j1b4

This parameter specifies the characteristics associated with each stream . 1j1b4a

stream-string is CHARSTR 1j1b5

This parameter is the stream to be written. 1j1b5a

stream-id-list is LIST(stream-id , ...) 1j1b6

This result list returns the appropriate stream-id for each individual parameter stream. 1j1b6a

read-stream (ATOM*, stream-id-list -> stream-parms-list) 1j2

This procedure returns the parameters associated with each stream specified in the stream-id-list. 1j2a

FORMAT: 1j2b

stream-id-list is LIST(stream-id , ...) 1j2b1

This parameter list specifies the stream-id for each stream parameter. 1j2b1a

stream-parms-list is LIST(stream, ...) 1j2b2

This parameter is a list of the parameters associated with each stream specified in the stream-id-list. 1j2b2a

stream is LIST(acoors, stream-parms) 1j2b3

This parameter specifies the individual characteristics of each stream. 1j2b3a

acoors is LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 1j2b4

This parameter specifies the positional coordinates of the stream relative to the atom origin. 1j2b4a

stream-parms is LIST(visibility, hit-sensitivity) 1j2b5

This parameter specifies the characteristics associated with each stream . 1j2b5a

manipulate-stream (ATOM*, stream-id-list, new-stream-parms-list) 1j3

This procedure permits the modification of the parameters associated with each stream specified in the stream-id-list. 1j3a

FORMAT: 1j3b

stream-id-list is LIST(stream-id , ...) 1j3b1

This parameter list specifies the stream-id for each new stream parameter. 1j3b1a

new-stream-parms-list is LIST(visibility, hit-sensitivity) 1j3b2

This parameter specifies the new parameters for each stream-id. 1j3b2a

1j4

Bug Control 1k

Send-Coors-with-action (FILE*, action-list) 1k1

Don't-send-Coors-with-actions (FILE*) 1k2

Report-Coors (FILE*, cursor-window-id, on-off, criteria-list) 1k3

Report-mouse-button-status (FILE*, criteria-list) 1k4

1k5

Bug Selection Manipulation 1l

Select-char(FILE*, coors -> window-id, atom-id, char-pos, wCoors) 1l1

This procedure accepts a FILE* and coordinates relative to the file and converts them to a window-id, atom-id, and line and character position within that string, and to coordinates relative to the selected window. Only strings that are hit sensitive will be considered as possible selection strings. 1l1a

FORMAT: 1l1b

coors = LIST(%x% INTEGER, %y% INTEGER) 1l1b1

char-pos = LIST(%line-number% INTEGER, %character-position% INTEGER) 1l1b2

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 111b3

Select-string(FILE*, coors -> window-id, atom-id, wcoors) 112

This procedure accepts a FILE* and coordinates relative to the file and converts them to a window-id and a atom-id, and to coordinates relative to the selected window. Only strings that are hit sensitive will be considered as possible selection strings. 112a

FORMAT: 112b

coors - LIST(%x% INTEGER, %y% INTEGER) 112b1

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 112b2

select-window(FILE*, coors -> window-id, wcoors) 113

This procedure accepts a FILE* and coordinates relative to the file and converts them to a window-id and to coordinates relative to the selected window. Only windows that are hit sensitive will be considered as possible selection windows. 113a

FORMAT: 113b

coors - LIST(%x% INTEGER, %y% INTEGER) 113b1

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 113b2

114

Bug Mark Manipulation 1m

mark-characters(WINDOW*, atom-id-1, string-pos-1, atom-id-2,
string-pos-2 -> mark-id) 1m1

This procedure will cause the (sub-)string to be made to "stand-out" in any appropriate terminal dependent manner. The mark-id returned can be used for future references to these characters that are now standing out. 1m1a

FORMAT: 1m1b

string-pos-1 - see string-pos above 1m1b1

string-pos-2 - see string-pos above 1m1b2

mark-id - INTEGER 1m1b3

remove-mark(FILE*, mark-id) 1m2

This procedure will cause characaters that were previously made to stand-out to no longer standout. 1m2a

FORMAT: 1m2b

mark-id - INTEGER / empty 1m2b1

If this parameter is empty, then all marked characters in the specified file will no longer standout. 1m2b1a

read-marks(FILE*, mark-id -> mark-id-list) 1m3

This procedure enables a process to determine which (sub-)strings are currently standing out. 1m3a

FORMAT: 1m3b

mark-id - INTEGER / empty 1m3b1

if this parameter is empty, then it refers to all marks for the indicated file. 1m3b1a

mark-id-list - LIST(LIST(mark-id, string-id, string-pos), ...) 1m3b2

1m4

Global Procedures 1n

process-batch(procedure-list) 1n1

This procedure allows a process to "batch" a group of procedure calls into one transmission. Any results from any of the procedures within the batch will be lost. 1n1a

FORMAT: 1n1b

procedure-list - LIST(LISI(pname, pargs), ...) 1n1b1

pname - INTEGER 1n1b2

the name of the procedure to be called 1n1b2a

pargs - any 1n1b3

these are the parameters to be passed to the procedure pname, 1n1b3a

1n2

JLE 2-MAY-75 13:00 25819

CLI - OSI Terminal Handler Specifications

(J25819) 2-MAY-75 13:00;;; Title: Author(s): JosePh L. Ehardt/JLE;
Distribution: /NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG;
Clerk: JOAN; Origin: < EHARDT, CLI-OSITH-WC.NLS;2, >, 17-FEB-75
14:14 JLE ;;;;
####;