NSW February Meeting Considerations

SCENARIOS 1

The following scenarios are in response to Larry Crain's memo announcing the Feb NSW review meeting. We have numbered the points we are addressing according to the numbers in Larry's original memo. 1a

3A) LOGIN AND LOGOUT 1b

This hasn't changed enough from CHI's earlier scenario (JOURNAL # 24534) to warrent much discussion. When a user types some character on an unused terminal, the FE collects project, username and password and calls login procedure in WM [We would write actual call here but dont have WM documentation]. The WM returns user-id, user profile for FE-interaction, and list of tools available to this user. User is then talking to NSW-EXEC grammar with commands to manipulate whole files, perform terminal-specific operations, get acounting information, logout, etc. In addittion the user always has available (while running any integrated tool) the universal commands to run tools, terminate tools, get semantic help with tools or the nSW as a whole. The number of commands in the universal set should be kept small to avoid undue restrictions on other tool command languages. 1b1

[Since FE has list of allowed tools, must it get permission from the WM before allowing user to run a tool?] 1b1a

3B) INVOKING, USING, AND LEAVING THE TELNET-ELF TOOL 1c

a) using ELF outside NSW 1c1

There will probably be a command in the NSW-EXEC that allows the user to leave the NSW FE and use the normal ELF exec. Once this is done, the user is on his own until he returns to the NSW FE. 1c1a

The user will not be able to reference NSW files by their NSW names. He will not be able to talk to the WM or NSW tools. 1c1a1

b) using a non-integrated tool 1c2

The NSW will allow users to use tools that are not fully integrated into the NSW. These tools will be accessed either a) through a common tool grammar that knows nothing of the behavior or intended function of the tool or b) through a tool grammar that has been tailored somewhat for that tool. 1c2a

In case (a) the user will type characters or strings to the tool and it will respond, with the FE doing all or no echoing. This will be much like operating a full-duplex or half-duplex character-at-a-time or a line-at-a-time terminal. There will be no commands given to the tool in the normal NSW sense of command words and parameters. The user will be able to get very little help from the FE for this type of tool since it has only one command which is just the collection of a literal string from the user, but he will have the universal commands available to him by typing an escape character. There will also be a command in the NSW-EXEC to allow the user to change his escape character. Please note that while running such a character-at-a-time tool, the normal characters for <back-space-character>, <bach-space-word>, <help>, etc. will not have their normal NSW function but will transmit that character to the un-integrated tool. Note also, that for line-at-a-time tools, the writer of the grammar may specify whether or not to send a carriage-return linefeed at the end of each string. 1c2b

In case (b) above, the tool grammar will contain commands tailored to the function of the tool and will appear to be more like an integrated tool. 1c2c

In both of these cases the NVT package will be used to drive the actual tool through telnet. The only difference is in the commands that are available to the user. In both cases the user may reference NSW files and may slue to other tools from the un-integrated one (see CHI's memo on tool interaction, 25120). 1c2d

The use of file names requires that the tool's attempt to access the file be trapped and that the file be moved to the local host by the WM. 1c2d1

3C) CREATING BATCH JOB 1d

This is covered in the NSWV2CHANGES file under the RJE-MODEL section. 1d1

3D) CALLING, USING, AND LEAVING NLS 1e

It should be understood that NLS like NSW represents a system for accessing a number of different tools. Thus, within the NSW the various tools contained in NLS will be tools in the NSW. There will be no single NLS tool. There will be an editor, a calculator, a send-mail, a user-profile tool, and perhaps other tools. 1e1

The universal command for running a tool is used to specify the desired tool, say the editor. 1e2

There is a tool naming issue here. We should not, for example, use up all of the obvious good names just because we are adding the first few tools. We propose that the user or his project leader supply the simple name which he will use and that this be translated into a unique system-wide name for the tool. Thus the user may ask to run the "editor" and for him that translates into "NLS-EDITOR." For another user, "editor" might mean some other editor tool. 1e2a

When the user logs into the NSW, the FE fetches from the WM a list of the tools this user is allowed to access. This list could consist of (simple name, system name) pairs. 1e2b

When the user issues the run-tool command he may type ? to find out which tools he may run. When the user specifies which tool is to be run, the FE calls the WM, passing it the (system) name of the tool and gets back the tool-id for this tool [is this necessary?]. If the grammar for the tool is already in the FE, then it is not reloaded. Otherwise, the FE calls the WM with the toolid and gets back the grammar for the tool. 1e3

we could implement this in such a way that the FE keeps track of tools used and does not bother to call the WM if this user has previously in this seession run this tool. As mentioned above, we could not bother the WM at all if the tool name is in the list of legal tools for this user. The WM can still stop a user from running a tool on a particular file since all file references must pass through the WM. 1e3a

The FE then inspects the grammar to determine which pcp process(es) must be created to support this tool. For each such process the WM is called to create it and introduce it to the FE. The FE opens the appropriate packages and allows the user to specify commands to the tool. 1e4

While the tool is being used, various procedures in the processes are called to carry out the semantics of the commands. 1e5

If the tool needs to read or write on a file it calls the WM to get the file. 1e6

While the user is using the tool, he may give a universal command such as run another tool or terminate the current tool. If he elects to run another tool without first terminating the current tool, the FE simply switches grammars and holds any

output from the old tool, the user may later terminate the new tool and resume the old tool or he may give the resume command for the old tool without terminating the new tool. This is what is meant by the term "slueing". When this happens, the FE switches back to the original grammar. 1e7

When the user terminates a tool, the WM is called to delete the process(es) that support this tool and the grammar's use count is decreased by one. if the use count is zero, then no user is using that tool and the core occupied by the grammar can be reclaimed if needed. 1e8

3E) CALLING FOR PROOFS, PUBLICATION TO COM 1f

A document has been entered into an NLS file and edited for content, spelling, grammar, etc. 1f1

The document is an Air Force 177 series manual in standard format and is to be produced, using COM, in both hardcopy and microfiche. 1f2

The user logs in to NSW and starts the NLS-Format tool. The Format grammer asks him to specify the name of the file to be formated, whether it is to be formated for COM or the line printer, and which of the standard formats to use. 1f3

The Frontend makes an out-of-line call on the Formater backend and the user is free to do other work while the formater inserts output processor directives in the file. 1f4

The user is notified when the process completes. 1f5

He may now examine the file containing directives, using the NLS-editor or immediately start the Output Processor tool. This tool produces two files: one is a sequential file, formated for a COM device to do the actual production of the document. The other is a file that serves as a page index both to the sequential COM file and the source file. In addition to pointers to the beginning and end of each page, the file contains the state information necessary to allow the output processor to start processing in the middle of a file. The pointers in this file are used to display formated pages on the graphics scope and to permit reprocessing of single or groups of pages from the source file. 1f6

Using the NLS-editor tool, the user may display his source file on the alphanumeric display and request the editor to display the COM formated version on the graphics display. 1f7

Viewing the COM formated document one page at a time, he may edit both text and directives in the source file. Hard copy proofs of all, or selected pages of the formated file may be made on the copy printer at the workstation. 1f8

When editing is finished, the user then processes those pages that have changed creating new sequential and pointer files. 1f9

When the output processor produces a satisfactory set of proofs, the works manager is used to transfer the sequential file to a tape at whatever host maintains contact wih the COM facility. (Note: this might not be an NSW host.) 1f10

3F) EXPLICIT (USER DIRECTED) FILE MOVEMENT INTO, OUT OF, AND WITHIN NSW 1g

This is accomplished via the NSW-EXEC's rename/copy/delete file commands. For copying files into and out of the NSW, the user must supply the necessary information to allow the file to be properly transfered and use-typed. 1g1

The FE will provide some abreviations for the local card reader, printer, and tape drive for use in these commands. If the file to be inserted into the NSW file system is online somewhere the user must supply the pathname to the file. 1g1a

We expect that the path names will look just like those used now in FTP. We also expect that MCA will provide procedures (in the WM or in a separate process) that are capable of talking old FTP and NSW file names (this could be done using the monitor call trapping mechanism for un-integrated tools). 1g1a1

It should be pointed out that we expect the WM to provide a file-name and file-name-field completion facility to the FE so that the user need only supply part of a file name and request the system to supply the rest for him (ala ESC and ^F in TENEX). 1g1b

In addition, we should state that since all tools must be able to refer file references to the WM, we see no value in the FE doing so also. Thus, we are not planning to report file references to the WM except, of course, as arguments to calls on WM procedures to support NSW-EXEC file commands, etc. 1g1c

We would also hope that the WM file system will provide the user with a facility like the MULTICS working directory or

the TENEX connected directory. If so, there will be a command in the NSW-EXEC to specify this. 1g1d

3G) HELP FEATURES 1h

This is accomplished via a universal command and keys on the user's terminal. 1h1

Keys: 1h1a

?: The user may type ? whenever specifying a command (except in the middle of literal text, of course). The FE responds with a list of current alternatives. 1h1a1

[We must decide what is meant by ? typed as the first character of a literal. Is the user asking what is wanted next or is the ? part of the literal text he is expected to type? We debated this for a long time for NLS-8 and finally decided to interpret it as a request for help. This occassionally causes a problem but it is easily understood by the user and happens rarely. If we use the other choice, the user will be unable to get help at times. This may be difficult to justify to the user, especially when he has several alternatives, only one of which is a literal.] 1h1a1a

SYNTAX: The user may type this key to learn the full syntax of a command, part of a command, or all commands in a tool. 1h1a2

HELP: The user may type this at any point in specifying a command to obtain semantic, functional help with the command, the tool containing the command, or with basic concepts in the NSW as a whole. 1h1a3

This is simply another way of accessing the semantic help facilities as described below. 1h1a3a

Command: 1h1b

The "HELP" command is in the universal commands and is thus available while using any integrated tool. It allows the user to specify a concept or command or a tool, etc, and attempts to provide the user with useful explanations thereof. The data base for this semantic help facility will be structured nls-editor files for first-year NSW. There will be one or more such files associated (by the WM or a declaration in the CML grammar) with each tool plus one or more containing

overall NSW concepts, lists of available tools, and guidelines for installing tools and tool help data bases. We are publishing guidelines for building such data bases. 1h1b1

[We should point out that it is not in our charter to supply the part of the data base describing the NSW as a whole, tools available within the NSW, and so forth. We strongly recommend that these exist but it is up to NSW management to charter and fund someone to supply these valuable aids to new users.] 1h1b1a

The process that interprets the structured data base and presents help to the user will be an instance of the nls-editor process, created at login time by the WM at the FE's request. When the user first requests semantic help this process is called with the name of the data base for the current tool. It obtains this file(s) plus the NSW-help file(s) from the WM and attempts to help the user. On subsequent invocations of the help facility, no new files will have to be obtained from the WM unless the user has switched tools. 1h1b2

Given our current model of how the help facility would work, it would be difficult for a user to find out detailed things about tools other than his current tool. We recommend that only an overview of other tools would be available to him. 1h1b2a

3H) INVOKING A TBH (TENEX, MULTICS, ?OS360/370) 1i

It is difficult for us to write a scenario about this since it violates our model of the NSW. The thing we think is implied here is starting a tool. It might mean starting a tool that is the interactive executive. 1i1

This should be no different than starting any other tool so the scenario should be the same as 3d (Calling using and leaving NLS). 1i2

3I) ESCAPING TO THE WM AND RETURNING TO A TOOL 1j

Escaping to the WM amounts to running the NSW-EXEC (this is done via a universal command or via an escape character). This "tool" is always immediattely available (the grammar is always in the sattelite machine and the WM process is always available). Once there the user may if he wishes suspend the current tool (in the middle of execution ala control-c in TENEX). We envision a "resume" command to be used to resume

such a suspended tool when the user wishes this to happen. If the tool being resumed was not suspended, but rather the user merely slued (via the escape-to-NSW-EXEC key, a "resume", or a "run" command) to another tool and is now sluing back, any output that was waiting for the user from the tool is now presented to him. 1j1

Following is a first pass at the set of universal commands and the commands in the NSW-EXEC: 1j2

universal commands 1j2a

run tool 1j2a1

("GOTO"!L2! (<tool-name>/"ELF"/"NSWEXEC") <confirm>) 1j2a1a

Note: The !L2! is CML notation to indicate that should the user request that frequently used commands be recognized based on their first letter, that this command will not be so recognized. It will require that the user type <space> before the command. This allows tools to have commands that begin with the same letter without causing a problem for such a user. If the user types a "g" in this case, he will get the tools command starting with "g", not the GOTO command. 1j2a1a1

terminate current tool 1j2a2

("QUIT"!L2! <confirm>) 1j2a2a

logout 1j2a3

("LOGOUT"!L2! <confirm>) 1j2a3a

resume tool 1j2a4

("RESUME"!L2! <tool-instance-name> <confirm>) 1j2a4a

execute command in another active tool 1j2a5

("EXECUTE"!L2! <tool-instance-name> <command>) 1j2a5a

comment 1j2a6

(";" <text> <confirm>) 1j2a6a

semantic help 1j2a7

("HELP" <optional-item-list> <confirm>) or 1j2a7a

Note: The "help" command will be recognized by its first letter for users who are using this type of recognition. This command is likely to be used by new users who may not understand the <space> to get to other commands. The system help facilities should always be readily available. 1j2a7a1

("<help>") 1j2a7b

show current commands 1j2a8

("?") 1j2a8a

show syntax of commands 1j2a9

("<syntax>") 1j2a9a

NSW-EXEC commands 1j2b

rename file 1j2b1

copy file 1j2b2

includes copying files into/out of the NSW. 1j2b2a

delete file 1j2b3

show 1j2b4

accounting info 1j2b4a

status of active tool(s) 1j2b4b

list of files 1j2b4c

working directory 1j2b4d

escape character 1j2b4e

set 1j2b5

working directory 1j2b5a

escape character 1j2b5b

tty window position and size (display terminal only) 1j2b5c

reset 1j2b6

working directory 1j2b6a

escape character 1j2b6b

tty window position and size (display terminal only) 1j2b6c

start/stop recording session (typescript) 1j2b7

playback session 1j2b8

connect/disconnect terminals 1j2b9

simulate terminal type 1j2b10

scroll back tty window (display only) 1j2b11

3J) PASSING MESSAGES IN NSW (NOT NLS JOURNAL OR NETMAIL) 1k

This will not happen. The only mechanism for user to exchange arbitrary text messages will be a mail tool either based on SNDMSG or the JOURNAL (most likely SNDMSG) with some interaction with a Works Manager maintained data base like an "Ident file". 1k1

3K) READING/SENDING JOURNAL NETWORK MAIL 1l

Sending a Letter Scenario
You have a CRT and line-processor console hooked up to the NSW. You want to compose and send a letter via U.S. mail to John. 1l1

Type gs. The words "Goto (subsystem) Sendmail" appear at the top of your screen in what is called the "command feedback line". You hit the CONFIRM key and type l (the letter). 1l1a

"Letter (Dear) T:" replaces the "Goto (subsystem) Sendmail" and you type John (the name of your recipient) and then the CONFIRM key. 1l1b

"(Body) C:" appears in the command feedback line. 1l1c

"C:" is a prompt for a command-word.
To discover what command-words are available, you hit the questionmark key. The screen contains the following words: 1l1c1

Current Alternatives are 111c1a

Branch, Group, File, Plex, Statement, or Text. 111c1a1

You type t. "Text B/T:" is appended to the command feedback line. Type the text of the business letter. The text appears on your video screen as you type it in. Use the key marked BC to backspace characters and the key marked BW to backspace words. You may type without worrying about the end of the line as new lines start automatically when needed. After you finish typing the paragraph, you hit two carriage returns. Your screen is cleared ready for the next paragraph. When you have finished typing the body of the letter, you hit the CONFIRM key. 111c2

"(Sincerely?) Y/N:" then appears in the command feedback line. You hit the CONFIRM key which means "Yes". You are sincere. Typing n would allow you to specify another closing. 111d

"(Author ident:) B/T:", appears. You type the author's NSW identifier. If you hit the NULL key, you are assumed to be the author. 111e

"(To) B/T:" appears. You type in John's name and address. If John had an ident, you could have typed it instead. Multiple mixed idents and addresses are also possible here and in the "Copies to" field which follows. Lists of idents cannot contain carriage returns and addresses must contain at least one carriage return and each address or group of idents must be seperated by double carriage returns as was done to terminate paragraphs above. 111f

"(Copies to) B/T:" appears. Although a copy will be kept for your records, you are not sending any copies to anyone so you hit the NULL key. 111g

"(Show Status?) Y/N/P/I:" appears. You type p CONFIRM for "Print" and your letter prints at your local printer along with all of your status information. The letter is formated containing a letter-head, heading, salutation, body, closing, and tracings. 111h

"(Send the Mail?) OK/C:" appears. 111i

You don't want to send it now because you notice a misspelled word in your letter. A questionmark shows you your 11111

Current alternatives are:
Delete (this letter)
Modify (the letter)
Sendmail (commands)
OK 111ii1a

You type m and CONFIRM. 111i2

"Modify (the letter) OK:" flashes by and your letter along with all of it's status information in a special, clearly marked form fills your screen. You are placed in an editor with which you can modify the status form. See Modifying a Document in the documentation production section. When you are done modifing, type q CONFIRM s CONFIRM. 111j

"Quit OK:" and "Send (the mail) OK:" appear. 111k

The letter prints at your local printer formated with the letter-head, heading, salutation, body, closing, and tracings. A separate page with John's address in the middle of the lower half and your address just below the middle against the left margin accompanies the letter. This can be folded in half over the letter page(s), stapled, stamped, and mailed. Or it can be cut out as a lable and pasted to a printed stamped envelope. 1111

If you specify that the letter is to be Archived before you say "Send the mail", a copy of the letter is stored in the computer which you can retrieve by its filename which is its NSW archive number. Also, a reference to the letter is placed in the list of Sendmail items you have authored. 111m

3L) GRAPHICS USER INTERFACES 1m

Graphics user interface takes three forms - user command set, virtual graphics interface, and physical graphics interface. The later two forms are further split into two sets, one for the data structure manipulation and the other for the terminal itself. 1mi

USER COMMAND SET 1m2

The user command set is the interface level which is utilized directly to manually create, view, and manipulate the diagrams stored with an NLS file. Generally speaking this interface takes two forms - manipulative commands, and drafting aids. 1m2a

Manipulative Commands 1m2a1

The commands in this class represent those used to create and modify a display. Since these commands are defined by the CML they can be easily tailored to user preference. While the exact command forms have not yet been formulated, this set of commands would include commands to: 1m2a1a

1) create and delete whole diagrams and to move them from one part of a file to another, or from file to file. 1m2a1a1

2) create, delete, and modify the atomic elements of a diagram, such as lines, curves, points, captions and text. 1m2a1a2

3) group collections of these atomic elements into structures for the creation of "templates" which can be stored and recalled; and for general modification of the diagram. For example, flowchart symbols would be constructed from the line and text elements, recalled with additional caption material, and added to the diagram being created. 1m2a1a3

Drafting Aids 1m2a2

Drafting aids include not only commands, but also environmental variables which constrain the cursor, provide scaling information, and aid the user in determining where a line or figure should be placed. For example, one command will set the resolution of the cursor, to effectively place a grid work over the screen so that alignment of figures within the drawing can be accomplished. 1m2a2a

VIRTUAL GRAPHICS INTERFACE 1m3

A virtual graphics interface will be needed to insure upward compatability with new graphics hardware (for example the moderate cost minicomputer based graphics terminal), and to provide the programmer with a consistant set of primative routines on which to base specific graphic user programs. 1m3a

PHYSICAL GRAPHICS INTERFACE 1m4

The virtual graphics interface will call the appropiate set of routines within the physical graphics interface. primitive calls in this group will maintain and move around within the NLS file system storing, modifying and retrieving

graphics data (as do the current text handling routines); and provide access to the physical graphics hardware (the routines which actually write on the display). 1m4a

The file system and the atomic elements for graphics are described in (belleville,nls-file-structure,). 1m4b

The physical terminal primatives for the initial (or minimum) graphics configuration are: 1m4c

gbegin() open the graphics port and stop tracking the cursor. 1m4c1

gend() resume cursor tracking. 1m4c2

gclear() clear the screen. 1m4c3

gline(pointer to linework data structure,pointer to coordinate transformation matrix,pointer to line type descriptor data structure) draw a line of the type specified using the given transformation. 1m4c4

gcap(pointer to a text data structure,pointer to a window,pointer to a transformation for the window,pointer to a text type (font etc) data structure) place the text within the window as given. 1m4c5

glccord(coordinate data structure) return the coordinates of the last cursor coordinate picked by the user at the terminal. (constrained by existing environment variables) 1m4c6

3M) DOCUMENT PRODUCTION SCENARIO 1n

You have a hand-written report with several sections that you wish to input on to a cassette, place into the NSW file system, correct any typographical errors, format adding a title-page, and print. You have only a typewriter-like terminal with a cassette device plugged into an NSW frontend computer. Note that the scenario using a CRT and line-processor instead would be considerably simpler than this. There are three major sections in this scenario, 1) typing onto the cassette tape and reading it into an NSW file, 2) viewing and editing the file online, 3) formatting and printing the edited file. 1n1

1) Typing onto a cassette tape to read in to an NSW file 1n2

preliminaries 1n2a

Switch on the typewriter terminal and the cassette device. Place a cassette in the cassette device. Be sure the cassette device is switched to "offline" so it is not talking to the computer. Type the keys on your terminal that cause the cassette tape to rewind and place the cassette device in record mode. 1n2a1

Type in the report 1n2b

Type the title of the report followed by a Carriage-Return (<CR>) and two Line Feeds (<LF> or <CTRL-J>). Do not bother with centering any titles. This can be done automatically later. 1n2b1

Type a lowercase d followed by a space and then "Section I". The d followed by a space indicates that Section I is to be located "down" under the title in the outline of the report. Type a <CR> and two <LF>s. 1n2b2

Type another d space followed by the first paragraph of Section I. 1n2b3

End every line with a <CR> and one <LF>. 1n2b4

End every paragraph and title with a <CR> and two <LF>s. 1n2b5

The lower case d space is not placed in front of the next paragraph because this and the following paragraphs in Section-I are at the same level in the outline of the report. 1n2b6

After ending the final paragraph in Section-I, type a lowercase u followed by a space and then "Section II". The u space indicates that Section II is located "up" at the same level as Section I in the outline of the report. 1n2b7

Type another d space and the first paragraph in Section II. 1n2b8

One d space is typed before each sub-heading and the first sub-paragraph after a heading. As many lower-case u's are entered as is necessary followed by a space to indicate the desired level of the next paragraph or heading. 1n2b9

This process is continued until the entire report has been entered. <CTRL-Z> is typed to indicate the end of the report. 1n2b10

Backspacing to correct errors while you type 1n2c

A cassette device usually has special keys you can hit to back space any number of characters, words, and up to the end of the last line. These edits are made on the cassette. In addition, you can type in any number or combination of < to backspace characters, > to backspace words, and ^ to backspace lines. These latter characters will be interpreted and the edits made when the information on the cassette is made into an NSW file of use-type NLS in the next step. 1n2c1

Creating a NSW File From a cassette Tape 1n2d

Switch the cassette machine off of record. Switch it online so it can talk to the NSW. Login to the NSW. You are automaticaly placed in your "login tool" which is the NLS editor. 1n2d1

Type gc. The words "Goto C: Cassette (tool) OK:" are echoed. You hit Carriage Return (<CR>) which means "OK". "CASS C:" is typed telling you that you have indeed gone to the Cassette reading tool and it is ready for you to specify a command-word. You type rd. 1n2d2

"Read C: Document (into file) T:" is echoed and you type the name of the report "july-report" followed by <CR>. This command creates an NSW file of use-type NLS. It looks for two carriage returns to specify the end of a paragraph or heading. 1n2d3

"Rewind tape? Y/N" appears and you type y for "yes". The tape then rewinds and the report is read into an NSW file. When it is finished reading the report, "(More?) Y/N:" is echoed. You answer n for "No". 1n2d4

Finally the terminal will type "CASS C:" indicating that the Cassette tool has finished reading the report into an NSW file and is ready for the next command. Type q <CR>. This returns you to the NLS editor. 1n2d5

2) Viewing and editing the file online

You have an NSW file named july-report which has been freshly input. You wish to proof-read it. You are logged into the NSW NLS editing tool. "EDIT C: " has been typed at your terminal indicating it is ready to receive NLS editing commands. 1n3

Type 1fju<ESC><CR>. "Load C: File T: juLY-REPORT," is

echoed on your terminal. The characters "JULY-REPORT" are also echoed on a separate line indicating that you have been placed at the beginning of the report. 1n3a

Type oq<CR>. "Output C: Quickprint OK/C:" is echoed. A copy of the report is printed on the local hardcopy printer specifed in your profile. The report has a number at the bottom right of each paragraph and heading uniquely representing it's position in the outline. Each paragraph is single spaced. There is one blank line separating each paragraph and title. These "viewspecs" are your default specified in your profile. 1n3b

While proof-reading the printed copy, you notice the same word is misspelled almost everywhere it occurs in the paper. 1n3c

Type swb0<CR>. "Substitute C: Word (in) OPT/C: Branch (at) A: 0" is echoed. Then in response to prompting from the command. You type the correct spelling followed by the incorrect spelling. When you are done, the words "25 substitutions made" are typed at your terminal. 1n3d

Further proofing reveals that the first paragraph in Section II (2A) should be moved after the last paragraph in Section I (1E). 1n3e

You type ms2a<CR>1d<CR><CR>. "Move C: Statement (from) 2a (to follow) 1d L: OK:" is echoed and you are ready for a new command. 1n3f

Noticing the word "can" is typed twice in a row in the paragraph in section III marked 3B, you type dw3b " can"<CR><CR>. "Delete C: Word (at) A: 3b " can" OK:" is echoed. When "EDIT C: " comes back indicating it is ready for a new command, you type the back-slash key \ and immediately, the paragraph you just edited is typed on your terminal. The word is gone. 1n3g

When you have completed all of the edits, you are ready to format the paper. For further information on NLS editing, see the TNLS-8 primer 23911, the NLS-8 Command Summary 23912, and the NLS-8 Glossary. 1n3h

3) Formatting and printing the edited file

You have loaded an NSW file name july-report which you wish to format and print on your local line-printer. 1n4

Type gf<CR>if<CR>3<CR>. 1n4a

EDIT C: Goto C: Format OK:
FORM C: Insert C: Format (in file at) A: (using format #)
3 1n4a1

is seen at your terminal followed by
"(Title:)" you type July Report<CR>
"(Author) Ident(s):)" you type the NSW identifiers of the authors. 1n4b

The Format tool then adds codes to the file to make the file conform to format number 3 which is the desired format for reports. It does such things as centering headings, adjusting margins, fixing type-font and size, and adding the title page. When "FORM C:" appears, you type q<CR> for "Quit OK:" and "EDIT: C: " is typed at your terminal. 1n4c

You type op<CR>. "Output C: Printer OK/C:" is echoed on your terminal and a formatted copy of the report is printed at your local line-printer. 1n4d

1n4e

THE HELP DESCRIPTION FILE 2

Background 2a

Most of the following background information is from 24485 "Some NSW Frontend Issues..." by Charles Irby 13-NOV-74 and 24534 "A Scenario of an NSW Session" by Charles Irby 17-NOV-74. 2a1

Typing the HELP button or using the Help command available for all tools can provide you, the user, with an English description from the current tool's Help description file(s) and place you in a repeating Help command. This will be accomplished by providing a separate function, capable of interacting with the user (via the Help command grammar in the Frontend) and using structured description files provided along with the tool grammar. This help function will not run in the satellite machine but will be invoked by the satellite whenever the user asks for semantic help with a tool. The help function will be provided with the name of the help description file(s) for the tool the user was using and a representation of the user's command state at the time he requested help. (Once a connection has been established to the help function for a user, the connection will probably be maintained until the user terminates the session.) 2a2

It is expected that the command language designers will provide the description files. It is expected that there will be one

description file for the NSW as a whole, describing global concepts, organization, purpose of the NSW. This description file will be available at all times to the user. In addition, we may wish to produce a description file that is a high-level guide or "yellow pages" to all the tools accessible through NSW. At any time the Universal description file(s) as well as the description file(s) for the tool currently being used are available. 2a3

SRI has not been funded to write and maintain the NSW description file(s) and we know of no one else who has been. There seems to be a hole here. 2a4

For first-year NSW, this help function is simply a set of calls on the NLS backend, with the description files being NLS structured files (this approach is now being used within NLS). 2a5

If the user requests semantic help with a tool the Frontend automatically starts the help function (which is probably loaded as needed rather than at Frontend startup time) and passes it information on the user's parse state, the name of the help description file(s) for this tool, the name of the NSW help description file, and the user-id so it can get at the user-profile. The user may interact with the help command for a while and then resume using the original tool. If he requests help again for the same tool, he merely switches to the help function which receives new parsestate info but otherwise preserves the state from the last interaction with this user. 2a6

The Help Command 2b

The following description of the Help command is adopted from the one in the NLS-8 description file. 2b1

HELP-button: <CTRL-Q>
Typing the HELP button (<CTRL-Q>) at any point in a command provides a description about what you were doing and places you in the Help command which allows you to ask for more information and the meanings of terms. 2b2

Help TYPEIN/OK:
The command "Help" provides the most complete information about a tool. After you type in any term and hit the Command Accept key (CA, <CTRL-D>), you will see the description. The Help command will be ready for another TYPEIN. TYPEIN any term you wish or the number of a "menu" followed by CA. Any time after the first description prints, you can type < followed by y (for yes) to see the previous view indicated or n (for no) to

choose a view before that. Hit the Command Delete key (CD, <CTRL-X>) to end the Help command. Capitalization does not matter when typing words in the Help command. 2b3

menu:
A numbered list of related subjects that may follow an explanation in the Help command. Typing a number followed by CA will show the explanation named. This list is called a menu. 2b3a

going-up (for advanced users): ^
If you use ^ instead of <, you will go "up" instead "back". Going up lets you "see your surroundings." Because of the "random access" nature of Help, it is sometimes the same but can be quite different from going back. This is just a convenience, it is not necessary for using the Help command. 2b3b

A Description 2c

A description consists of an NLS statement containing a short paragraph. The first word of a paragraph can be made the "name" of that paragraph and is the term defined by that paragraph. Users of the Help command can get any description simply by typing the term. Provisions exist for using multiple words to specify duplicate terms within the same description file. 2c1

Menued paragraphs are numbered sub-paragraphs classified by the term in the paragraph under which they are located in the outline or tree-structure of the file. Only the first line of menued items appear until they have been requested by typing the corresponding "menu number". 2c2

A paragraph may consist of a term, some optional supporting words, and a pointer or "link" to another paragraph in the current description file or any NLS file. If descriptions are written properly, you can avoid much redundancy by linking from one concept to another. The description file containing links takes on the qualities of a network. If it is well structured, it becomes a hierarchical network. 2c3

Structuring a Description File 2d

Depending on the tool, description file structure will vary. At the minimum, there must be a description of the tool in general terms. A list of descriptions of the commands available in the tool with names the same as each command word must exist in order for the HELP button to find and display

them. These are placed under the general tool description in the file structure. Commands with a tree structure of alternative command words may need a corresponding tree structure in the description file describing the alternatives. 2d1

Usually, there are a few command functions which occur in many commands. These may be given names and described in only one place. In addition, step-by-step scenarios of how to do specific tasks that can be accomplished with the tool may be provided. These are written in words the user can understand which interface the user to confusing or criptic commands. Besides pointing or "linking" to the desired commands, these "How to" descriptions can be structured to present any special terms the user needs to learn in the most effective way. 2d2

If "How to" descriptions are provided, they are usually listed in an appropriate order terminated by the branch containing all of the command descriptions. This "command description branch" starts with a statement named "commands" which appears as the last menu when reading the general, top-level description of the tool. 2d3

The NSW description File(s) 2e

The NSW description file(s) will contain descriptions of all of the commands in the Works Manager (WM), and the Front End (FE), the NSW-EXEC and Universal commands. Some subset of these commands will be "Universal" commands available to all tools that are integrated into the NSW. 2e1

In addition, any high-level concepts and definitions of terms necessary to use the WM, the FE, and the NSW in general should be available here. This can include general descriptions of tools or, to avoid duplication, links to tool description files. The various tools can be placed under subject headings and indexes to the terms used in the description files of each tool can be provided thus making up the "yellow pages" of the NSW. Such links to description files can be followed using the Help command if the access controls allow it. In the future, it may be desireable due to the simplicity of the Help command to actually startup a tool in this manner. 2e2

We know of no one funded to write any of the NSW description file(s). 2e3

Helpd: Proposed Help Description File Development Tool 2f

We recommend that in the second year of NSW a Help description file development tool be built. The purpose of this tool would

be to help create, maintain and publish a tool's dexcription file. The tool would not only prompt a tool builder for commands and Help descriptions, but would also perform verification of the links and structure in his Help data bases. 2f1

IMPORTANT AREAS FOR DISCUSSION 3

FILENAME 3a

The NSW filenameing convention used by the NSW Works Manager (WM) will differ in significant ways from both TENEX and current NLS filenameing conventions. One of the ARC goals is to ensure a consistent user-interaction across tools integrated into the NSW. This means the same convention should be used for naming files in all NSW tools. In keeping with this philosophy, the NLS-9 filenaming convention should match the convention used by the WM. As NSW users, front-end builders, and tool-integrators we want the filename syntax to be the easiest to type and point to, the most flexible in use, the fastest to parse, and the least offensive to look at. 3a1

Speed of parsing a filename is a major point of difference between NSW and the current NLS. We do not want to burden the NSW with the current baroque NLS-8 link parser. Delimiters around the filename and a place for an infile-address within those delimiters, on the other hand, are two features potentially valuable to any tool and should be carried over from the current NLS-8 into NLS-9. Such a delimited "address" or "path-name" containing a filename, infile-address, or both imbedded in text is called a "link". A third field of a link, the viewspecs, have been treated in discussions about this as a part of the infile-address only. 3a2

Delimiters around the filename are needed for ease of pointing to a filename imbedded in text. In general, as the detail or number of selections increases, the effort necessary to select increases geometrically. The easiest way to specify something is to name what type it is (e.g. a link) using the appropriate command and then make one specification near enough to the item in front or in back of it to distinguish it from others of it's kind and have the command find it and grab it. In order to do this, the item must be enclosed in "enclosing" delimiters which are available on all terminals. In addition, the delimiters must not be common characters that might be usefull in a filename or outside a filename. Since parentheses are frequently used for parenthetical expressions, and square- and squigly-brackets are not on all terminals, that leaves only angle-brackets. We therefore recommend that angle-brackets be

the NSW filename or link delimiter. It turns out that this is also an acceptable delimiter for current NLS links. 3a3

The infile-address needs to be within the same delimiters as the filename because it is an integral part of the entire path-name or address of which the filename may be only the beginning. Tool builders that allow an infile-address will want to use the same delimiters for links that do not happen to go across files. A single reserved separater character is necessary to distinguish a filename field preceeding an infile-address field so that each field may contain the maximum range of characters. The separater should be easy to type because unlike the delimiters for a link, the separater may be frequently typed by the user as free text in a command. The only easily typed punctuation characters are period, comma, slash, and semi-colon with period and comma probably the easiest. Our experience with NLS has shown that comma works very well. For maximum compatability and minimum conversion hassle we recommend that comma be the separater character. 3a4

For speed and accuracy of parsing, we do not want to allow the delimiters inside the delimiters and we do not want to allow the separating character in the filename or infile-address fields. 3a5

filename = ['<] filename [, infile-address] ['>]
filename and infile-address do not include '<, '>, ',. 3a5a

Note: We should point out that when a user types a link or filename he need not type the angle brackets as the Frontend will provide these for him. 3a5b

JBP's description of Bob Millstein's syntax for NSW filenames 25205 looks like it would fit our needs described above. One character substitution and the addition of the possibility of delimiters also containing infile-addresses would be necessary. We request that comma not be used anywhere in the filename. Charles points out that there should be no reason why fields in a filename can't be seperated by a simple space rather than a somewhat more ugly punctuation character. For somthing that is the least offensive to look at, this would be desireable. 3a6

One unanswered question is "What does the link <abc> point to -- a filename or an infile-address?" It is clear that <abc,> always points to a filename and <,abc> always points to an infile-address because filenames always preceed the infile-address. 3a7

The current default in NLS is for a link containing no comma

to be taken as an infile-address. In the initial NSW, it may be that a link containing only a filename will be more frequent than a link containing only an infile-address and we should therefore switch defaults. 3a7a

In links, this would require a comma at the beginning of every infile-address but not at the end of a filename with no infile-address. 3a7b

The TNLS user using an infile-address to specify locations in editing commands will not want to place a comma in front of every address. A special function would be written to not require it at that point so that infile-address specification would be the same as NLS-8. This would mean the user must always after a filename when prompted by A: but need not place a comma when using a "file" command (such as Load File) or in links. 3a7c

A link to filename abc may look like <abc> but if viewspecs vspc are specified, it must look like <abc, :vspc>. 3a7d

A fancy infile-address parser would be necessary to allow commas in content searches, otherwise characters preceeding the comma would be mistaken for a filename. If ' or " are allowed in filenames, searches for commas may be unparseable. 3a7e

The infile-address should allow constructs such as "..." and 'char so that content addressing may include the literal characters "," and ">". 3a7f

Samples of links containing only an infile address are <,,abc> and <,,#abc>. Those containing only viewspecs would look like <,,:vspc>. 3a7g

The question of whether or not ARC should recommend deviating from its current default has not been decided. 3a7h

USER PROFILE 3b

This section outlines the current design of "User Profiles" as used by both the Front End and the NLS tools within the NSW environment. The Works Manager functions needed to support this design are also detailed. 3b1

Overview 3b2

An NSW user wants to have control over some of the parameters which control the interaction between himself and

the NSW system. The FE must have access to a file, or a data store which defines the user's interaction parameters. 3b2a

The first question to be decided is whether a "user Profile" is bound to an individual , or to an individual,project pair, that is to an account. It seems more consistent with the overall goals of the NSW to have at least part of the "user profile" bound to an individual, regardless of which project he is currently working on. We envision the FE making use of such an "individual profile" to control the interaction between the Command Language Interpreter (CLI) and the individual. This includes such things as command recognition mode, prompting mode, and the verboseness to be used. 3b2b

Elements in a user's profile which describe his access rights, however must clearly be based on the account, that is on the user-project pair. We are assuming that the WM will provide both a grammar and its supporting packages to maintain these data bases. We would like the FE to be able to read a part of the account profile data maintained by the WM at login time. This allows the FE to provide some useful functions for the user. For example suppose that at login time the FE hands the WM the user name and project identifier and recieves in return a list of the tools that the user can use. This enables the FE to provide a reasonable reply when the user types "RUN (tool) ?". The FE reponds with the list of tools that are available to the user. Another example might be a data element called entry tool. If the WM can provide the FE with this data element for a user-project pair the FE can place the user directly into this tool after login. 3b2c

In addition each NSW tool may require it's own elements of user profile data which are completely independent of the FE and WM. NLS for example contains the address of a commands branch to be processed upon entry to NLS, and a link file to be used to resolve external names in a jump command. 3b2d

It seems unreasonable to require the WM to maintain any tool dependent user profiles, or to even know of there existence. It should clearly be the burden of the tool manufacturer to mainatin any tool dependent user profile for his tool. This can be done by either including the appropiate profile modifing commands in his tool, or by providing a separate tool which maintains the user profile. Note that even though the WM is not directly involed in this maintainence the actual user profile data base has to be a NSW file, that is

known to the WM in order to provide host independence to the tool. 3b2e

Recommendation 3b3

The NSW FE will make use of two profile data bases. One is called the "individual" profile and the other is the "tool" profile. ARC will provide the grammar and the backend process to maintain a users individual profile. We request that the WM makes primitives available to the FE read the elements of the tool profile from the WM's account profile. The following is a list of the data elements which we think would be good candidates for elements in the FE tool profile. 3b3a

List of "approved tools" 3b3a1

Entry tool 3b3a2

In this model the WM has the following responsiblities concerning user profiles. 3b3b

The works manager will provide a grammar and supporting process which maintains the account profile for each user, project pair. It is probable that use of this facility will be restricted to project leaders. 3b3b1

Primitives will be made available to the FE for reading agreed parts of this account profile, namely the tool profile. 3b3b2

In NSW the NLS tool will keep its own user profile (individual profile) for each user. The grammar will contain the proper commands for modification of the data elements . These commands will be supported by a package in the NLS Back End. 3b3c

To implement a single user profile for an individual it is necessary that the Works Manager provide a unique identifier for each NSW individual. A later section will discuss the need for, and possible designs of such a unique identifier. Basically what is required is a WM primitive which will take as arguments a user name and project name and return a unique identifier for this individual. Note that the process which maintains the FE's individual profile also requires this primitive. 3b3d

Requested WM primitives : 3b3e

available tools: 3b3e1

availtools(username,project => toollist, entrytool) 3b3e1a

This primitive will be called by the FE to build a tool profile for this user, for this session. 3b3e1b

Argument / result types 3b3e1c

username - CHARSTR 3b3e1c1

project - CHARSTR 3b3e1c2

toollist - LIST (%toolnames% (simplename, systemname)...) 3b3e1c3

entrytool - INTEGER/EMPTY 3b3e1c4

unique user identifier: 3b3e2

uniqueid(username,project => userid) 3b3e2a

This primitive is called by the tool which maintains the users individual profile , and also by the FE to get a handle on this individual profile. Some tools may also use this primitive. 3b3e2b

Argument / result types 3b3e2c

username - CHARSTR 3b3e2c1

project - CHARSTR 3b3e2c2

userid - LIST (INTEGER,CHRSTR) 3b3e2c3

IDENT SYSTEM 3c

The NSW needs to be able to deliver mail for an individual to a single mail box and to know the type of delivery the individual would like, i.e. an NLS-JOURNAL citation or a 'SNDMSG' sequential file. 3c1

In addition to mail delivery we should anticipate the need for NSW directories and 'phone' books. 3c2

The NLS editing tool needs an identifier for an individual. We presently have available 21 bits that can be translated to a displayable, meaningful, character string to use in statement signatures (simple audit trails). 3c3

NSW February Meeting Considerations

In the current NLS we provide the necessary information in a special file that contains the following information. 3c4

Individuals 3c4a

Information needed for mail delivery 3c4a1

Name: two fields, lastname, first and middle
This allows us to deal with split names like van Kamp.
Ident : a 4 character alpha numeric identifier or nickname
Organization (see below)
Hardcopy mail address
Network mail address: host name
Delivery mode: Hardcopy / Network Sequential / Network NLS 3c4a1a

Additional Information for Directories (Phone Books, etc.) 3c4a2

Phones
Groups: Idents of all the groups the person belongs to
Function
Capabilities
Secondary organization
Comments
Subcollections: Used for indexing 3c4a2a

Groups 3c4b

Information needed for mail delivery 3c4b1

Name
Ident
Membership: The Idents of all members
Hardcopy mail address
Network mail address
Delivery
Coordinator 3c4b1a

Additional Information for Directories (Phone Books, etc.) 3c4b2

Function
Comments 3c4b2a

Organizations(Projects) 3c4c

Information needed for mail delivery 3c4c1

NSW February Meeting Considerations

Name
Ident
Membership
Groups
Coordinator
Hardcopy mail address
Network mail address
Delivery 3c4c1a

Additional Information for Directories (Phone Books, etc.) 3c4c2

Type of organization
Phone
Comments 3c4c2a

The 4 character ident has not been fully satisfactory as duplications occur frequently, requiring idents such as RLB2. However, our present file format limits us to 21 bits for the identifier. We suggust using a 21 bit permanent number that can be translated to a character string to use both in statement signatures and as a query argument. 3c5

By permanent we mean that the number, sequentially assigned shall never be reused. 3c6

In addition to the number each record should contain a permanent ident (nickname), limited to, say 50 (upper case ?) printing characters. Each inidvidual would choose his own ident. 3c7

Consideration should be given to other information which might be useful. 3c8

It is particularly important that a super fast search across this file be possible. 3c9

In addition to providing the mail tool with its needs, the database should be queryable by people. Minimal query arguments should include ident (nickname) and last name. 3c10

We can see three possible ways of dealing with this for the first year of NSW. These are 3c11

Find a way to get BBN TIPSER DAtabse right for NSW needs 3c11a

Include all the needed information in the Works Manager's data base. 3c11b

During the first year use the NLS ident system for mailing.

The main problem with this is the 4 character limit on nicknames. 3c11c

More Questions: 3c12

What does the Works Manager know about real people? 3c12a

How does a tool ask the WM for information about people? 3c12b

What does the WM return in response to an inquiry. 3c12c

Who maintains the data base? i.e, who can enter, and validate the information in the file. We see this as a big, on-going problem area. 3c12d

NPG 3-FEB-75 22:21 25289

NSW February Meeting Considerations

(J25289) 3-FEB-75 22:21;;;; Title: Author(s): Joe L. Ehardt, Robert Louis Belleville, Robert Louis Belleville, Jonathan B. Postel, Kirk E. Kelley, Karolyn J. Martin, David S. Maynard, Kenneth E. (Ken) Victor, James E. (Jim) White, Elizabeth K. Michael, Don I. Andrews, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman/NPG; Distribution: /NPG([ACTION]) RWW([ACTION]) EKM([INFO-ONLY]) ; Sub-Collections: NPG; Clerk: EKM; Origin: < MICHAEL, JANSCEN.NLS;8, >, 3-FEB-75 22:00 EKM ;;;;####;

(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)252
(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)252
(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)25289	(EKM)252

TUESDAY, FEBRUARY 4, 1975 15:27:20-PST TUESDAY, FEBRUARY 4, 1975 15:27:20-PS
TUESDAY, FEBRUARY 4, 1975 15:27:20-PST TUESDAY, FEBRUARY 4, 1975 15:27:20-PS
TUESDAY, FEBRUARY 4, 1975 15:27:20-PST TUESDAY, FEBRUARY 4, 1975 15:27:20-PS

Return to DVN

<HJOURNAL>25289.NLS:1, 4-FEB-75 04:34 XXX ;;;: .HJOURNAL="NPG 3-FEB-75 22:21 25289": Title: .H1="NSW February Meeting Considerations"; Author(s): Joe L. Enardt, Robert Louis Belleville, Robert Louis Belleville, Jonathan B. Postel, Kirk E. Kelley, Karolyn J. Martin, David S. Maynard, Kenneth E. (Ken) Victor, James E. (Jim) White, Elizabeth K. Michael, Don I. Andrews, J. D. Hopper, Charles H. Irby, Harvey G. Lehtman/NPG: Distribution: /NPG(/ ACTION /) RWW(/ ACTION /) EKM(/ INFO-ONLY /) : Sub-collections: NPG; Clerk: EKM; .IGD=0; .SNF=HJRM: .RM=HJRM-7: .PN=-1; .YBS=1; .PES; Origin: < MICHAEL, JANSCEN.NLS:8, >, 3-FEB-75 22:00 EKM :;:;####:

SCENARIOS 1

The following scenarios are in response to Larry Crain's memo announcing the Feb NSW review meeting. We have numbered the points we are addressing according to the numbers in Larry's original memo. 1A 1B

3A) LOGIN AND LOGOUT

This hasn't changed enough from CHI's earlier scenario (JOURNAL # 21534) to warrent much discussion. When a user types some character on an unused terminal, the FE collects project, username and password and calls login procedure in WM /We would write actual call here but dont have WM documentation/. The WM returns user-id, user profile for FE-interaction, and list of tools available to this user. User is then talking to NSW-EXEC grammar with commands to manipulate whole files, perform terminal-specific operations, get acounting information, logout, etc. In addittion the user always has available (while running any integrated tool) the universal commands to run tools, terminate tools, get semantic help with tools or the nSW as a whole. The number of commands in the universal set should be kept small to avoid undue restrictions on other tool command languages. 1B1

/ Since FE has list of allowed tools, must it get permission from the WM before allowing user to run a tool? / 1B1A

3B) INVOKING, USING, AND LEAVING THE TELNET-ELF TOOL 1C

a) using ELF outside NSW 1C1

There will probably be a command in the NSW-EXEC that allows the user to leave the NSW FE and use the normal ELF exec. Once this is done, the user is on his own until he returns to the NSW FE. 1C1A

The user will not be able to reference NSW files by their NSW names. He will not be able to talk to the WM or NSW tools. 1C1A1

b) using a non-integrated tool 1C2

The NSW will allow users to use tools that are not fully

integrated into the NSW. These tools will be accessed either a) through a common tool grammar that knows nothing of the behavior or intended function of the tool or b) through a tool grammar that has been tailored somewhat for that tool. 1C2A

In case (a) the user will type characters or strings to the tool and it will respond, with the FE doing all or no echoing. This will be much like operating a full-duplex or half-duplex character-at-a-time or a line-at-a-time terminal. There will be no commands given to the tool in the normal NSW sense of command words and parameters. The user will be able to get very little help from the FE for this type of tool since it has only one command which is just the collection of a literal string from the user, but he will have the universal commands available to him by typing an escape character. There will also be a command in the NSW-EXEC to allow the user to change his escape character. Please note that while running such a character-at-a-time tool, the normal characters for <back-space-character>, <each-space-word>, <help>, etc. will not have their normal NSW function but will transmit that character to the un-integrated tool. Note also, that for line-at-a-time tools, the writer of the grammar may specify whether or not to send a carriage-return linefeed at the end of each string. 1C2B

In case (b) above, the tool grammar will contain commands tailored to the function of the tool and will appear to be more like an integrated tool. 1C2C

In both of these cases the NVT package will be used to drive the actual tool through telnet. The only difference is in the commands that are available to the user. In both cases the user may reference NSW files and may slue to other tools from the un-integrated one (see CHI's memo on tool interaction, 25120). 1C2D

The use of file names requires that the tool's attempt to access the file be trapped and that the file be moved to the local host by the WM. 1C2D1

3C) CREATING BATCH JOB 1D

This is covered in the NSWV2CHANGES file under the RJE-MODEL section. 1D1

3D) CALLING, USING, AND LEAVING NLS 1E

It should be understood that NLS like NSW represents a system for accessing a number of different tools. Thus, within the NSW the various tools contained in NLS will be tools in the NSW. There will be no single NLS tool. There will be an editor, a calculator, a send-mail, a user-profile tool, and perhaps other tools. 1E1

The universal command for running a tool is used to specify the desired tool, say the editor. 1E2

There is a tool naming issue here. We should not, for example, use up all of the obvious good names just because we are adding the first few tools. We propose that the user or his project leader supply the simple name which he will use and that this be translated into a unique system-wide name for the tool. Thus the user may ask to run the "editor" and for him that translates into "NLS-EDITOR." For another user, "editor" might mean some other editor tool. 1E2A

Uses def for Help

When the user logs into the NSW, the FE fetches from the WM a list of the tools this user is allowed to access. This list could consist of (simple name, system name) pairs. 1E2B

When the user issues the run-tool command he may type ? to find out which tools he may run. When the user specifies which tool is to be run, the FE calls the WM, passing it the (system) name of the tool and gets back the tool-id for this tool (is this necessary?). If the grammar for the tool is already in the FE, then it is not reloaded. Otherwise, the FE calls the WM with the toolid and gets back the grammar for the tool. 1E3

We could implement this in such a way that the FE keeps track of tools used and does not bother to call the WM if this user has previously in this seession run this tool. As mentioned above, we could not bother the WM at all if the tool name is in the list of legal tools for this user. The WM can still stop a user from running a tool on a particular file since all file references must pass through the WM. 1E3A

The FE then inspects the grammar to determine which pcp process(es) must be created to support this tool. For each such process the WM is called to create it and introduce it to the FE. The FE opens the appropriate packages and allows the user to specify commands to the tool. 1E4

While the tool is being used, various procedures in the processes are called to carry out the semantics of the commands. 1E5

If the tool needs to read or write on a file it calls the WM to get the file. 1E6

While the user is using the tool, he may give a universal command such as run another tool or terminate the current tool. If he elects to run another tool without first terminating the current tool, the FE simply switches grammars and holds any output from the old tool. the user may later terminate the new tool and resume the old tool or he may give the resume command for the old tool without terminating the new tool. This is what is meant by the term "slueing". When this happens, the FE switches back to the original grammar. 1E7

When the user terminates a tool, the WM is called to delete the process(es) that support this tool and the grammar's use count is decreased by one. if the use count is zero, then no user is using that tool and the core occupied by the grammar can be

reclaimed if needed. 1E8

3E) CALLING FOR PROOFS, PUBLICATION TO COM 1F

A document has been entered into an NLS file and edited for content, spelling, grammar, etc. 1F1

The document is an Air Force 177 series manual in standard format and is to be produced, using COM, in both hardcopy and microfiche. 1F2

The user logs in to NSW and starts the NLS-Format tool. The Format grammer asks him to specify the name of the file to be formated, whether it is to be formated for COM or the line printer, and which of the standard formats to use. 1F3

The Frontend makes an out-of-line call on the Formater backend and the user is free to do other work while the formater inserts output processor directives in the file. 1F4

The user is notified when the process completes. 1F5

He may now examine the file containing directives, using the NLS-editor or immediately start the Output Processor tool. This tool produces two files: one is a sequential file, formated for a COM device to do the actual production of the document. The other is a file that serves as a page index both to the sequential COM file and the source file. In addition to pointers to the beginning and end of each page, the file contains the state information necessary to allow the output processor to start processing in the middle of a file. The pointers in this file are used to display formated pages on the graphics scope and to permit reprocessing of single or groups of pages from the source file. 1F6

Using the NLS-editor tool, the user may display his source file on the alphanumeric display and request the editor to display the COM formated version on the graphics display. 1F7

Viewing the COM formated document one page at a time, he may edit both text and directives in the source file. Hard copy proofs of all, or selected pages of the formated file may be made on the copy printer at the workstation. 1F8

When editing is finished, the user then processes those pages that have changed creating new sequential and pointer files. 1F9

When the output processor produces a satisfactory set of proofs, the works manager is used to transfer the sequential file to a tape at whatever host maintains contact wih the COM facility. (Note: this might not be an NSW host.) 1F10

3F) EXPLICIT (USER DIRECTED) FILE MOVEMENT INTO, OUT OF, AND WITHIN NSW 1G

This is accomplished via the NSW-EXEC's rename/copy/delete file

commands. For copying files into and out of the NSW, the user must supply the necessary information to allow the file to be properly transfered and use-typed. 1G1

The FE will provide some abreviations for the local card reader, printer, and tape drive for use in these commands. If the file to be inserted into the NSW file system is online somewhere the user must supply the pathname to the file. 1G1A

We expect that the path names will look just like those used now in FTP. We also expect that MCA will provide procedures (in the WM or in a separate process) that are capable of talking old FTP and NSW file names (this could be done using the monitor call trapping mechanism for un-integrated tools). 1G1A1

It should be pointed out that we expect the WM to provide a file-name and file-name-field completion facility to the FE so that the user need only supply part of a file name and request the system to supply the rest for him (ala ESC and ↑F in TENEX). 1G1B

In addition, we should state that since all tools must be able to refer file references to the WM, we see no value in the FE doing so also. Thus, we are not planning to report file references to the WM except, of course, as arguments to calls on WM procedures to support NSW-EXEC file commands, etc. 1G1C

We would also hope that the WM file system will provide the user with a facility like the MULTICS working directory or the TENEX connected directory. If so, there will be a command in the NSW-EXEC to specify this. 1G1D

3G) HELP FEATURES 1H

This is accomplished via a universal command and keys on the user's terminal. 1H1

Keys: 1H1A

?: The user may type ? whenever specifying a command (except in the middle of literal text, of course). The FE responds with a list of current alternatives. 1H1A1

/We must decide what is meant by ? typed as the first character of a literal. Is the user asking what is wanted next or is the ? part of the literal text he is expected to type? We debated this for a long time for NLS-8 and finally decided to interpret it as a request for help. This occassionally causes a problem but it is easily understood by the user and happens rarely. If we use the other choice, the user will be unable to get help at times. This may be difficult to justify to the user, especially when he has several alternatives, only one of which is a literal./ 1H1A1A

Not much real, maybe we don't need it for stray leads

SYNTAX: The user may type this key to learn the full syntax of a command, part of a command, or all commands in a tool. 1H1A2

HELP: The user may type this at any point in specifying a command to obtain semantic, functional help with the command, the tool containing the command, or with basic concepts in the NSW as a whole. 1H1A3

This is simply another way of accessing the semantic help facilities as described below. 1H1A3A

Command: 1H1B

The "HELP" command is in the universal commands and is thus available while using any integrated tool. It allows the user to specify a concept or command or a tool, etc. and attempts to provide the user with useful explanations thereof. The data base for this semantic help facility will be structured nls-editor files for first-year NSW. There will be one or more such files associated (by the WM or a declaration in the CML grammar) with each tool plus one or more containing overall NSW concepts, lists of available tools, and guidelines for installing tools and tool help data bases. We are publishing guidelines for building such data bases. 1H1B1

[We should point out that it is not in our charter to supply the part of the data base describing the NSW as a whole, tools available within the NSW, and so forth. We strongly recommend that these exist but it is up to NSW management to charter and fund someone to supply these valuable aids to new users.] 1H1B1A

The process that interprets the structured data base and presents help to the user will be an instance of the nls-editor process, created at login time by the WM at the FE's request. When the user first requests semantic help this process is called with the name of the data base for the current tool. It obtains this file(s) plus the NSW-help file(s) from the WM and attempts to help the user. On subsequent invocations of the help facility, no new files will have to be obtained from the WM unless the user has switched tools. 1H1B2

Given our current model of how the help facility would work, it would be difficult for a user to find out detailed things about tools other than his current tool. We recommend that only an overview of other tools would be available to him. 1H1B2A

3H) INVOKING A TBH (TENEX, MULTICS, ?OS360/370) 1I

It is difficult for us to write a scenario about this since it violates our model of the NSW. The thing we think is implied here is starting a tool. It might mean starting a tool that is the interactive executive. 1I1

This should be no different than starting any other tool so the scenario should be the same as 3d (calling using and leaving NLS). 1I2

3I) ESCAPING TO THE WM AND RETURNING TO A TOOL 1J

Escaping to the WM amounts to running the NSW-EXEC (this is done via a universal command or via an escape character). This "tool" is always immediattely available (the grammar is always in the sattelite machine and the WM process is always available). Once there the user may if he wishes suspend the current tool (in the middle of execution ala control-c in TENEX). We envision a "resume" command to be used to resume such a suspended tool when the user wishes this to happen. If the tool being resumed was not suspended, but rather the user merely slued (via the escape-to-NSW-EXEC key, a "resume", or a "run" command) to another tool and is now sluing back, any output that was waiting for the user from the tool is now presented to him. 1J1

Following is a first pass at the set of universal commands and the commands in the NSW-EXEC: 1J2

universal commands 1J2A

run tool 1J2A1

("GOTO"!L2! (<tool-name>/"ELF"/"NSWEXEC") <confirm>) 1J2A1A

Note: The !L2! is CML notation to indicate that should the user request that frequently used commands be recognized based on their first letter, that this command will not be so recognized. It will require that the user type <space> before the command. This allows tools to have commands that begin with the same letter without causing a problem for such a user. If the user types a "g" in this case, he will get the tools command starting with "g", not the GOTO command. 1J2A1A1

terminate current tool 1J2A2

("QUIT"!L2! <confirm>) 1J2A2A

logout 1J2A3

("LOGOUT"!L2! <confirm>) 1J2A3A

resume tool 1J2A4

("RESUME"!L2! <tool-instance-name> <confirm>) 1J2A4A

execute command in another active tool 1J2A5

("EXECUTE"!L2! <tool-instance-name> <Command>) 1J2A5A

comment 1J2A6

(":" <text> <confirm>) 1J2A6A

semantic help 1J2A7

("HELP" <optional-item-list> <confirm>) or 1J2A7A

Note: The "help" command will be recognized by its first letter for users who are using this type of recognition. This command is likely to be used by new users who may not understand the <space> to get to other commands. The system help facilities should always be readily available. 1J2A7A1

("<help>") 1J2A7B

show current commands 1J2A8

("?") 1J2A8A

show syntax of commands 1J2A9

("<syntax>") 1J2A9A

NSW-EXEC commands 1J2B

rename file 1J2B1

copy file 1J2B2

includes copying files into/out of the NSW. 1J2B2A

delete file 1J2B3

show 1J2B4

accounting info 1J2B4A

status of active tool(s) 1J2B4B

list of files 1J2B4C

working directory 1J2B4D

escape character 1J2B4E

set 1J2B5

working directory 1J2B5A

escape character 1J2B5B

tty window position and size (display terminal only) 1J2B5C

reset 1J2B6

working directory 1J2B6A

escape character 1J2B6B

tty window position and size (display terminal only) 1J2B6C

start/stop recording session (typescript) 1J2B7

playback session 1J2B8

connect/disconnect terminals 1J2B9

simulate terminal type 1J2B10

scroll back tty window (display only) 1J2B11

3J) PASSING MESSAGES IN NSW (NOT NLS JOURNAL OR NETMAIL) 1K

This will not happen. The only mechanism for user to exchange arbitrary text messages will be a mail tool either based on SNDMSG or the JOURNAL (most likely SNDMSG) with some interaction with a Works Manager maintained data base like an "Ident file". 1K1

3K) READING/SENDING JOURNAL NETWORK MAIL 1L

Sending a Letter Scenario
You have a CRT and line-processor console hooked up to the NSW. You want to compose and send a letter via U.S. mail to John. 1L1

Type gs. The words "Goto (subsystem) Sendmail" appear at the top of your screen in what is called the "command feedback line". You hit the CONFIRM key and type l (the letter). 1L1A

"Letter (Dear) T:" replaces the "Goto (subsystem) Sendmail" and you type John (the name of your recipient) and then the CONFIRM key. 1L1B

"(Body) C:" appears in the command feedback line. 1L1C

"C:" is a prompt for a command-word.
To discover what command-words are available, you hit the questionmark key. The screen contains the following words: 1L1C1

Current Alternatives are 1L1C1A

Branch, Group, File, Plex, Statement, or Text. 1L1C1A1

You type t. "Text B/T:" is appended to the command feedback line. Type the text of the business letter. The text appears on your video screen as you type it in. Use the key marked BC to backspace characters and the key marked BW to backspace words. You may type without worrying about the end of the line as new lines start

automatically when needed. After you finish typing the paragraph, you hit two carriage returns. Your screen is cleared ready for the next paragraph. When you have finished typing the body of the letter, you hit the CONFIRM key. 1L1C2

"(Sincerely?) Y/N:" then appears in the command feedback line. You hit the CONFIRM key which means "Yes". You are sincere. Typing n would allow you to specify another closing. 1L1D

"(Author ident:) B/T:", appears. You type the author's NSW identifier. If you hit the NULL key, you are assumed to be the author. 1L1E

"(To) B/T:" appears. You type in John's name and address. If John had an ident, you could have typed it instead. Multiple mixed idents and addresses are also possible here and in the "Copies to" field which follows. Lists of idents cannot contain carriage returns and addresses must contain at least one carriage return and each address or group of idents must be seperated by double carriage returns as was done to terminate paragraphs above. 1L1F

"(Copies to) B/T:" appears. Although a copy will be kept for your records, you are not sending any copies to anyone so you hit the NULL key. 1L1G

"(Show Status?) Y/N/P/I:" appears. You type p CONFIRM for "Print" and your letter prints at your local printer along with all of your status information. The letter is formated containing a letter-head, heading, salutation, body, closing, and tracings. 1L1H

"(Send the Mail?) OK/C:" appears. 1L1I

You don't want to send it now because you notice a misspelled word in your letter. A questionmark shows you your 1L1I1

Current alternatives are:
Delete (this letter)
Modify (the letter)
Sendmail (commands)
OK 1L1I1A

You type m and CONFIRM. 1L1I2

"Modify (the letter) OK:" flashes by and your letter along with all of it's status information in a special, clearly marked form fills your screen. You are placed in an editor with which you can modify the status form. See Modifying a Document in the documentation production section. When you are done modifing, type q CONFIRM s CONFIRM. 1L1J

"Quit OK:" and "Send (the mail) OK:" appear. 1L1K

The letter prints at your local printer formated with the letter-head, heading, salutation, body, closing, and tracings. A separate page with John's address in the middle of the lower half and your address just below the middle against the left margin accompanies the letter. This can be folded in half over the letter page(s), stapled, stamped, and mailed. Or it can be cut out as a lable and pasted to a printed stamped envelope. 1L1L

If you specify that the letter is to be Archived before you say "Send the mail", a copy of the letter is stored in the computer which you can retrieve by its filename which is its NSW archive number. Also, a reference to the letter is placed in the list of Sendmail items you have authored. 1L1M

3L) GRAPHICS USER INTERFACES 1M

Graphics user interface takes three forms - user command set, virtual graphics interface, and physical graphics interface. The later two forms are further split into two sets, one for the data structure manipulation and the other for the terminal itself. 1M1

USER COMMAND SET 1M2

The user command set is the interface level which is utilized directly to manually create, view, and manipulate the diagrams stored with an NLS file. Generally speaking this interface takes two forms - manipulative commands, and drafting aids. 1M2A

Manipulative Commands 1M2A1

The commands in this class represent those used to create and modify a display. Since these commands are defined by the CML they can be easily tailored to user preference. While the exact command forms have not yet been formulated, this set of commands would include commands to: 1M2A1A

1) create and delete whole diagrams and to move them from one part of a file to another, or from file to file. 1M2A1A1

2) create, delete, and modify the atomic elements of a diagram, such as lines, curves, points, captions and text. 1M2A1A2

3) group collections of these atomic elements into structures for the creation of "templates" which can be stored and recalled; and for general modification of the diagram. For example, flowchart symbols would be constructed from the line and text elements, recalled with additional caption material, and added to the diagram being created. 1M2A1A3

Drafting Aids 1M2A2

Drafting aids include not only commands, but also environmental variables which constrain the cursor, provide scaling information, and aid the user in determining where a line or figure should be placed. For example, one command will set the resolution of the cursor, to effectively place a grid work over the screen so that alignment of figures within the drawing can be accomplished. 1M2A2A

VIRTUAL GRAPHICS INTERFACE 1M3

A virtual graphics interface will be needed to insure upward compatability with new graphics hardware (for example the moderate cost minicomputer based graphics terminal), and to provide the programmer with a consistant set of primative routines on which to base specific graphic user programs. 1M3A

PHYSICAL GRAPHICS INTERFACE 1M4

The virtual graphics interface will call the appropiate set of routines within the physical graphics interface. Primitive calls in this group will maintain and move around within the NLS file system storing, modifying and retrieving graphics data (as do the current text handling routines); and provide access to the physical graphics hardware (the routines which actually write on the display). 1M4A

The file system and the atomic elements for graphics are described in (belleville,nls-file-structure,). 1M4B

The physical terminal primatives for the initial (or minimum) graphics configuration are: 1M4C

gbegin() open the graphics port and stop tracking the cursor. 1M4C1

gend() resume cursor tracking. 1M4C2

gclear() clear the screen. 1M4C3

gline(pointer to linework data structure,pointer to coordinate transformation matrix,pointer to line type descriptor data structure) draw a line of the type specified using the given transformation. 1M4C4

gcap(pointer to a text data structure,pointer to a window,pointer to a transformation for the window,pointer to a text type (font etc) data structure) place the text within the window as given. 1M4C5

gliccord(coordinate data structure) return the coordinates of the last cursor coordinate picked by the user at the terminal. (constrained by existing environment variables) 1M4C6

3M) DOCUMENT PRODUCTION SCENARIO 1N

You have a hand-written report with several sections that you wish to input on to a cassette, place into the NSW file system, correct any typographical errors, format adding a title-page, and print. You have only a typewriter-like terminal with a cassette device plugged into an NSW frontend computer. Note that the scenario using a CRT and line-processor instead would be considerably simpler than this. There are three major sections in this scenario. 1) typing onto the cassette tape and reading it into an NSW file, 2) viewing and editing the file online, 3) formatting and printing the edited file. 1N1

1) Typing onto a cassette tape to read in to an NSW file 1N2

Preliminaries 1N2A

Switch on the typewriter terminal and the cassette device. Place a cassette in the cassette device. Be sure the cassette device is switched to "offline" so it is not talking to the computer. Type the keys on your terminal that cause the cassette tape to rewind and place the cassette device in record mode. 1N2A1

Type in the report 1N2B

Type the title of the report followed by a Carriage-Return (<CR>) and two Line Feeds (<LF> or <CTRL-J>). Do not bother with centering any titles. This can be done automatically later. 1N2B1

Type a lowercase d followed by a space and then "Section I". The d followed by a space indicates that Section I is to be located "down" under the title in the outline of the report. Type a <CR> and two <LF>s. 1N2B2

Type another d space followed by the first paragraph of Section I. 1N2B3

End every line with a <CR> and one <LF>. 1N2B4

End every paragraph and title with a <CR> and two <LF>s. 1N2B5

The lower case d space is not placed in front of the next paragraph because this and the following paragraphs in Section-I are at the same level in the outline of the report. 1N2B6

After ending the final paragraph in Section-I, type a lowercase u followed by a space and then "Section II". The u space indicates that Section II is located "up" at the same level as Section I in the outline of the report. 1N2B7

Type another d space and the first paragraph in Section II. 1N2B8

One d space is typed before each sub-heading and the first sub-paragraph after a heading. As many lower-case u's are entered as is necessary followed by a space to indicate the

desired level of the next paragraph or heading. 1N2B9

This process is continued until the entire report has been entered. <CTRL-Z> is typed to indicate the end of the report. 1N2B10

Backspacing to correct errors while you type 1N2C

A cassette device usually has special keys you can hit to back space any number of characters, words, and up to the end of the last line. These edits are made on the cassette. In addition, you can type in any number or combination of < to backspace characters, > to backspace words, and ↑ to backspace lines. These latter characters will be interpreted and the edits made when the information on the cassette is made into an NSW file of use-type NLS in the next step. 1N2C1

Creating a NSW File From a cassette Tape 1N2D

Switch the cassette machine off of record. Switch it online so it can talk to the NSW. Login to the NSW. You are automaticaly placed in your "login tool" which is the NLS editor. 1N2D1

Type gc. The words "Gcto C: Cassette (tool) OK:" are echoed. You hit Carriage Return (<CR>) which means "OK". "CASS C:" is typed telling you that you have indeed gone to the Cassette reading tool and it is ready for you to specify a command-word. You type rd. 1N2D2

"Read C: Document (into file) T:" is echoed and you type the name of the report "july-report" followed by <CR>. This command creates an NSW file of use-type NLS. It looks for two carriage returns to specify the end of a paragraph or heading. 1N2D3

"Rewind tape? Y/N" appears and you type y for "yes". The tape then rewinds and the report is read into an NSW file. When it is finished reading the report, "(More?) Y/N:" is echoed. You answer n for "No". 1N2D4

Finally the terminal will type "CASS C:" indicating that the Cassette tool has finished reading the report into an NSW file and is ready for the next command. Type q <CR>. This returns you to the NLS editor. 1N2D5

2) Viewing and editing the file online

You have an NSW file named july-report which has been freshly input. You wish to proof-read it. You are logged into the NSW NLS editing tool. "EDIT C: " has been typed at your terminal indicating it is ready to receive NLS editing commands. 1N3

Type lfju<ESC><CR>. "Load C: File T: JULY-REPORT," is echoed on your terminal. The characters "JULY-REPORT" are also echoed on a separate line indicating that you have been placed

at the beginning of the report. 1N3A

Type oq<CR>. "Output C: Quickprint OK/C:" is echoed. A copy of the report is printed on the local hardcopy printer specified in your profile. The report has a number at the bottom right of each paragraph and heading uniquely representing it's position in the outline. Each paragraph is single spaced. There is one blank line separating each paragraph and title. These "viewspecs" are your default specified in your profile. 1N3B

While proof-reading the printed copy, you notice the same word is misspelled almost everywhere it occurs in the paper. 1N3C

Type swbO<CR>. "Substitute C: Word (in) OPT/C: Branch (at) A: O" is echoed. Then in response to prompting from the command, You type the correct spelling followed by the incorrect spelling. When you are done, the words "25 substitutions made" are typed at your terminal. 1N3D

Further proofing reveals that the first paragraph in Section II (2A) should be moved after the last paragraph in Section I (1E). 1N3E

You type ms2a<CR>1d<CR><CR>. "Move C: Statement (from) 2a (to follow) 1d L: OK:" is echoed and you are ready for a new command. 1N3F

Noticing the word "can" is typed twice in a row in the paragraph in section III marked 3B, you type dw3b " can"<CR><CR>. "Delete C: Word (at) A: 3b " can" OK:" is echoed. When "EDIT C: " comes back indicating it is ready for a new command, you type the back-slash key \ and immediately, the paragraph you just edited is typed on your terminal. The word is gone. 1N3G

When you have completed all of the edits, you are ready to format the paper. For further information on NLS editing, see the TNLS-8 primer 23911, the NLS-8 Command Summary 23912, and the NLS-8 Glossary. 1N3H

3) Formatting and printing the edited file

You have loaded an NSW file name july-report which you wish to format and print on your local line-printer. 1N4

Type gf<CR>if<CR>3<CR>. 1N4A

EDIT C: Goto C: Format OK:
FORM C: Insert C: Format (in file at) A: (using format #) 3
1N4A1

is seen at your terminal followed by
"(Title:)" you type July Report<CR>
"(Author) Ident(s):)" you type the NSW identifiers of the authors. 1N4B

The Format tool then adds codes to the file to make the file

conform to format number 3 which is the desired format for reports. It does such things as centering headings, adjusting margins, fixing type-font and size, and adding the title page. When "FORM C:" appears, you type q<CR> for "Quit OK:" and "EDIT: C: " is typed at your terminal. 1N4C

You type op<CR>. "Output C: Printer OK/C:" is echoed on your terminal and a formatted copy of the report is printed at your local line-printer. 1N4D

1N4E

THE HELP DESCRIPTION FILE 2

Background 2A

Most of the following background information is from 24485 "Some NSW Frontend Issues..." by Charles Irby 13-NOV-74 and 24534 "A Scenario of an NSW Session" by Charles Irby 17-NOV-74. 2A1

Typing the HELP button or using the Help command available for all tools can provide you, the user, with an English description from the current tool's Help description file(s) and place you in a repeating Help command. This will be accomplished by providing a separate function, capable of interacting with the user (via the Help command grammar in the Frontend) and using structured description files provided along with the tool grammar. This help function will not run in the satellite machine but will be invoked by the satellite whenever the user asks for semantic help with a tool. The help function will be provided with the name of the help description file(s) for the tool the user was using and a representation of the user's command state at the time he requested help. (Once a connection has been established to the help function for a user, the connection will probably be maintained until the user terminates the session.) 2A2

It is expected that the command language designers will provide the description files. It is expected that there will be one description file for the NSW as a whole, describing global concepts, organization, purpose of the NSW. This description file will be available at all times to the user. In addition, we may wish to produce a description file that is a high-level guide or "yellow pages" to all the tools accessible through NSW. At any time the Universal description file(s) as well as the description file(s) for the tool currently being used are available. 2A3

SRI has not been funded to write and maintain the NSW description file(s) and we know of no one else who has been. There seems to be a hole here. 2A4

For first-year NSW, this help function is simply a set of calls on the NLS backend, with the description files being NLS structured files (this approach is now being used within NLS). 2A5

If the user requests semantic help with a tool the Frontend automatically starts the help function (which is probably loaded as needed rather than at Frontend startup time) and passes it information on the user's parse state, the name of the help description file(s) for this tool, the name of the NSW help description file, and the user-id so it can get at the user-profile. The user may interact with the help command for a while and then resume using the original tool. If he requests help again for the same tool, he merely switches to the help function which receives new parsestate info but otherwise preserves the state from the last interaction with this user. 2A6

The Help Command 2B

The following description of the Help command is adopted from the one in the NLS-8 description file. 2B1

HELP-button: <CTRL-Q>
Typing the HELP button (<CTRL-Q>) at any point in a command provides a description about what you were doing and places you in the Help command which allows you to ask for more information and the meanings of terms. 2B2

Help TYPEIN/OK:
The command "Help" provides the most complete information about a tool. After you type in any term and hit the Command Accept key (CA, <CTRL-D>), you will see the description. The Help command will be ready for another TYPEIN. TYPEIN any term you wish or the number of a "menu" followed by CA. Any time after the first description prints, you can type < followed by y (for yes) to see the previous view indicated or n (for no) to choose a view before that. Hit the Command Delete key (CD, <CTRL-X>) to end the Help command. Capitalization does not matter when typing words in the Help command. 2B3

menu:
A numbered list of related subjects that may follow an explanation in the Help command, Typing a number followed by CA will show the explanation named. This list is called a menu. 2B3A

going-up (for advanced users): ↑
If you use ↑ instead of <, you will go "up" instead "back". Going up lets you "see your surroundings." Because of the "random access" nature of Help, it is sometimes the same but can be quite different from going back. This is just a convenience, it is not necessary for using the Help command. 2B3B

A Description 2C

A description consists of an NLS statement containing a short paragraph. The first word of a paragraph can be made the "name" of that paragraph and is the term defined by that paragraph. Users of the Help command can get any description simply by typing the term. Provisions exist for using multiple words to specify duplicate terms within the same description file. 2C1

Menued paragraphs are numbered sub-paragraphs classified by the term in the paragraph under which they are located in the outline or tree-structure of the file. Only the first line of menued items appear until they have been requested by typing the corresponding "menu number". 2C2

A paragraph may consist of a term, some optional supporting words, and a pointer or "link" to another paragraph in the current description file or any NLS file. If descriptions are written properly, you can avoid much redundancy by linking from

one concept to another. The description file containing links takes on the qualities of a network. If it is well structured, it becomes a hierarchical network. 2C3

Structuring a Description File 2D

Depending on the tool, description file structure will vary. At the minimum, there must be a description of the tool in general terms. A list of descriptions of the commands available in the tool with names the same as each command word must exist in order for the HELP button to find and display them. These are placed under the general tool description in the file structure. Commands with a tree structure of alternative command words may need a corresponding tree structure in the description file describing the alternatives. 2D1

Usually, there are a few command functions which occur in many commands. These may be given names and described in only one place. In addition, step-by-step scenarios of how to do specific tasks that can be accomplished with the tool may be provided. These are written in words the user can understand which interface the user to confusing or criptic commands. Besides pointing or "linking" to the desired commands, these "How to" descriptions can be structured to present any special terms the user needs to learn in the most effective way. 2D2

If "How to" descriptions are provided, they are usually listed in an appropriate order terminated by the branch containing all of the command descriptions. This "command description branch" starts with a statement named "commands" which appears as the last menu when reading the general, top-level description of the tool. 2D3

The NSW description File(s) 2E

The NSW description file(s) will contain descriptions of all of the commands in the Works Manager (WM), and the Front End (FE), the NSW-EXEC and Universal commands. Some subset of these commands will be "Universal" commands available to all tools that are integrated into the NSW. 2E1

In addition, any high-level concepts and definitions of terms necessary to use the WM, the FE, and the NSW in general should be available here. This can include general descriptions of tools or, to avoid duplication, links to tool description files. The various tools can be placed under subject headings and indexes to the terms used in the description files of each tool can be provided thus making up the "yellow pages" of the NSW. Such links to description files can be followed using the Help command if the access controls allow it. In the future, it may be desireable due to the simplicity of the Help command to actually startup a tool in this manner. 2E2

We know of no one funded to write any of the NSW description file(s). 2E3

Helpd: Proposed Help Description File Development Tool 2F

We recommend that in the second year of NSW a Help description file development tool be built. The purpose of this tool would be to help create, maintain and publish a tool's dexcription file. The tool would not only prompt a tool builder for commands and Help descriptions, but would also perform verification of the links and structure in his Help data bases. 2F1

IMPORTANT AREAS FOR DISCUSSION 3

FILENAME 3A

The NSW filenameing convention used by the NSW Works Manager (WM) will differ in significant ways from both TENEX and current NLS filenameing conventions. One of the ARC goals is to ensure a consistent user-interaction across tools integrated into the NSW. This means the same convention should be used for naming files in all NSW tools. In keeping with this philosophy, the NLS-9 filenaming convention should match the convention used by the WM. As NSW users, front-end builders, and tool-integrators we want the filename syntax to be the easiest to type and point to, the most flexible in use, the fastest to parse, and the least offensive to look at. 3A1

Speed of parsing a filename is a major point of difference between NSW and the current NLS. We do not want to burden the NSW with the current baroque NLS-8 link parser. Delimiters around the filename and a place for an infile-address within those delimiters, on the other hand, are two features potentially valuable to any tool and should be carried over from the current NLS-8 into NLS-9. Such a delimited "address" or "path-name" containing a filename, infile-address, or both imbedded in text is called a "link". A third field of a link, the viewspecs, have been treated in discussions about this as a part of the infile-address only. 3A2

Delimiters around the filename are needed for ease of pointing to a filename imbedded in text. In general, as the detail or number of selections increases, the effort necessary to select increases geometrically. The easiest way to specify something is to name what type it is (e.g. a link) using the appropriate command and then make one specification near enough to the item in front or in back of it to distinguish it from others of it's kind and have the command find it and grab it. In order to do this, the item must be enclosed in "enclosing" delimiters which are available on all terminals. In addition, the delimiters must not be common characters that might be usefull in a filename or outside a filename. Since parentheses are frequently used for parenthetical expressions, and square- and squigly-brackets are not on all terminals, that leaves only angle-brackets. We therefore recommend that angle-brackets be the NSW filename or link delimiter. It turns out that this is also an acceptable delimiter for current NLS links. 3A3

The infile-address needs to be within the same delimiters as the

filename because it is an integral part of the entire path-name or address of which the filename may be only the beginning. Tool builders that allow an infile-address will want to use the same delimiters for links that do not happen to go across files. A single reserved separater character is necessary to distinguish a filename field preceeding an infile-address field so that each field may contain the maximum range of characters. The separater should be easy to type because unlike the delimiters for a link, the separater may be frequently typed by the user as free text in a command. The only easily typed punctuation characters are period, comma, slash, and semi-colon with period and comma probably the easiest. Our experience with NLS has shown that comma works very well. For maximum compatability and minimum conversion hassle we recommend that comma be the separater character. 3A4

For speed and accuracy of parsing, we do not want to allow the delimiters inside the delimiters and we do not want to allow the separating character in the filename or infile-address fields. 3A5

filename = ['<] filename [, infile-address] ['>]
filename and infile-address do not include '<, '>, ',. 3A5A

Note: We should point out that when a user types a link or filename he need not type the angle brackets as the Frontend will provide these for him. 3A5B

JBP's description of Bob Millstein's syntax for NSW filenames 25205 looks like it would fit our needs described above. One character substitution and the addition of the possibility of delimiters also containing infile-addresses would be necessary. We request that comma not be used anywhere in the filename. Charles points out that there should be no reason why fields in a filename can't be seperated by a simple space rather than a somewhat more ugly punctuation character. For somthing that is the least offensive to look at, this would be desireable. 3A6

One unanswered question is "What does the link <abc> point to -- a filename or an infile-address?" It is clear that <abc,> always points to a filename and <,abc> always points to an infile-address because filenames always preceed the infile-address. 3A7

The current default in NLS is for a link containing no comma to be taken as an infile-address. In the initial NSW, it may be that a link containing only a filename will be more frequent than a link containing only an infile-address and we should therefore switch defaults. 3A7A

In links, this would require a comma at the beginning of every infile-address but not at the end of a filename with no infile-address. 3A7B

The TNLS user using an infile-address to specify locations in editing commands will not want to place a comma in front of every address. A special function would be written to not

require it at that point so that infile-address specification would be the same as NLS-8. This would mean the user must always after a filename when prompted by A: but need not place a comma when using a "file" command (such as Load File) or in links. 3A7C

A link to filename abc may look like <abc> but if viewspecs vspc are specified, it must look like <abc, :vspc>. 3A7D

A fancy infile-address parser would be necessary to allow commas in content searches, otherwise characters preceeding the comma would be mistaken for a filename. If ' or " are allowed in filenames, searches for commas may be unparseable. 3A7E

The infile-address should allow constructs such as "...." and 'char so that content addressing may include the literal characters "," and ">". 3A7F

Samples of links containing only an infile address are <,.abc> and <,#abc>. Those containing only viewspecs would look like <,:vspc>. 3A7G

The question of whether or not ARC should recommend deviating from its current default has not been decided. 3A7H

USER PROFILE 3B

This section outlines the current design of "User Profiles" as used by both the Front End and the NLS tools within the NSW environment. The Works Manager functions needed to support this design are also detailed. 3B1

Overview 3B2

An NSW user wants to have control over some of the parameters which control the interaction between himself and the NSW system. The FE must have access to a file, or a data store which defines the user's interaction parameters. 3B2A

The first question to be decided is whether a "user Profile" is bound to an individual , or to an individual,project pair, that is to an account. It seems more consistent with the overall goals of the NSW to have at least part of the "user profile" bound to an individual, regardless of which project he is currently working on. We envision the FE making use of such an "individual profile" to control the interaction between the Command Language Interpreter (CLI) and the individual. This includes such things as command recognition mode, prompting mode, and the verboseness to be used. 3B2B

Elements in a user's profile which describe his access rights, however must clearly be based on the account, that is on the user-project pair. We are assuming that the WM will provide both a grammar and its supporting packages to maintain these data bases. We would like the FE to be able to read a part of the account profile data maintained by the WM at login time.

This allows the FE to provide some useful functions for the user. For example suppose that at login time the FE hands the WM the user name and project identifier and recieves in return a list of the tools that the user can use. This enables the FE to provide a reasonable reply when the user types "RUN (tool) ?". The FE reponds with the list of tools that are available to the user. Another example might be a data element called entry tool. If the WM can provide the FE with this data element for a user-project pair the FE can place the user directly into this tool after login. 3B2C

In addition each NSW tool may require it's own elements of user profile data which are completely independent of the FE and WM. NLS for example contains the address of a commands branch to be processed upon entry to NLS, and a link file to be used to resolve external names in a jump command. 3B2D

It seems unreasonable to require the WM to maintain any tool dependent user profiles, or to even know of there existence. It should clearly be the burden of the tool manufacturer to mainatin any tool dependent user profile for his tool. This can be done by either including the appropiate profile modifing commands in his tool, or by providing a separate tool which maintains the user profile. Note that even though the WM is not directly involed in this maintainence the actual user profile data base has to be a NSW file, that is known to the WM in order to provide host independence to the tool. 3B2E

Recommendation 3B3

The NSW FE will make use of two profile data bases. One is called the "individual" profile and the other is the "tool" profile. ARC will provide the grammar and the backend process to maintain a users individual profile. We request that the WM makes primitives available to the FE read the elements of the tool profile from the WM's account profile. The following is a list of the data elements which we think would be good candidates for elements in the FE tool profile. 3B3A

List of "approved tools" 3B3A1

Entry tool 3B3A2

In this model the WM has the following responsiblities concerning user profiles. 3B3B

The works manager will provide a grammar and supporting process which maintains the account profile for each user, project pair. It is probable that use of this facility will be restricted to project leaders. 3B3B1

Primitives will be made available to the FE for reading agreed parts of this account profile, namely the tool profile. 3B3B2

In NSW the NLS tool will keep its own user profile (individual

profile) for each user. The grammar will contain the proper commands for modification of the data elements . These commands will be supported by a package in the NLS Back End. 3B3C

To implement a single user profile for an individual it is necessary that the Works Manager provide a unique identifier for each NSW individual. A later section will discuss the need for, and possible designs of such a unique identifier. Basically what is required is a WM primitive which will take as arguments a user name and project name and return a unique identifier for this individual. Note that the process which maintains the FE's individual profile also requires this primitive. 3B3D

Requested WM primitives : 3B3E

available tools: 3B3E1

availtools(username,project -> toollist, entrytool) 3B3E1A

This primitive will be called by the FE to build a tool profile for this user, for this session. 3B3E1B

Argument / result types 3B3E1C

username - CHARSTR 3B3E1C1

project - CHARSTR 3B3E1C2

toollist - LIST (%toolnames% (simplename, systemname)...) 3B3E1C3

entrytool - INTEGER/EMPTY 3B3E1C4

unique user identifier: 3B3E2

uniqueid(username,project -> userid) 3B3E2A

This primitive is called by the tool which maintains the users individual profile , and also by the FE to get a handle on this individual profile. Some tools may also use this primitive. 3B3E2B

Argument / result types 3B3E2C

username - CHARSTR 3B3E2C1

project - CHARSTR 3B3E2C2

userid - LIST (INTEGER,CHRSTR) 3B3E2C3

IDENT SYSTEM 3C

The NSW needs to be able to deliver mail for an individual to a single mail box and to know the type of delivery the individual would like, i.e. an NLS-JOURNAL citation or a 'SNDMSG' sequential

file. 3C1

In addition to mail delivery we should anticipate the need for NSW directories and 'phone' books. 3C2

The NLS editing tool needs an identifier for an individual. We presently have available 21 bits that can be translated to a displayable, meaningful, character string to use in statement signatures (simple audit trails). 3C3

In the current NLS we provide the necessary information in a special file that contains the following information. 3C4

Individuals 3C4A

Information needed for mail delivery 3C4A1

Name: two fields, lastname, first and middle
This allows us to deal with split names like van Kamp.
Ident : a 4 character alpha numeric identifier or nickname
Organization (see below)
Hardcopy mail address '
Network mail address: host name
Delivery mode: Hardcopy / Network Sequential / Network NLS 3C4A1A

Additional Information for Directories (Phone Books, etc.) 3C4A2

Phones
Groups: Idents of all the groups the person belongs to
Function
Capabilities
Secondary organization
Comments
Subcollections: Used for indexing 3C4A2A

Groups 3C4B

Information needed for mail delivery 3C4B1

Name
Ident
Membership: The Idents of all members
Hardcopy mail address
Network mail address
Delivery
Coordinator 3C4B1A

Additional Information for Directories (Phone Books, etc.) 3C4B2

Function
Comments 3C4B2A

Organizations(Projects) 3C4C

Information needed for mail delivery 3C4C1

Name
Ident
Membership
Groups
Coordinator
Hardcopy mail address
Network mail address
Delivery 3C4C1A

Addiional Information for Directories (Phone Books, etc.) 3C4C2

Type of organization
Phone
Comments 3C4C2A

The 4 character ident has not been fully satisfactory as duplications occur frequently, requiring idents such as RLB2. However, our present file format limits us to 21 bits for the identifier. We suggust using a 21 bit permanent number that can be translated to a character string to use both in statement signatures and as a query argument. 3C5

By permanent we mean that the number, sequentially assigned shall never be reused. 3C6

In addition to the number each record should contain a permanent ident (nickname), limited to, say 50 (upper case ?) printing characters. Each inidvidual would choose his own ident. 3C7

Consideration should be given to other information which might be useful. 3C8

It is particularly important that a super fast search across this file be possible. 3C9

In addition to providing the mail tool with its needs, the database should be queryable by people. Minimal query arguments should include ident (nickname) and last name. 3C10

We can see three possible ways of dealing with this for the first year of NSW. These are 3C11

Find a way to get BBN TIPSER DAtabse right for NSW needs 3C11A

Include all the needed information in the Works Manager's data base. 3C11B

During the first year use the NLS ident system for mailing. The main problem with this is the 4 character limit on nicknames. 3C11C

More Questions: 3C12

What does the Works Manager know about real people? 3C12A

How does a tool ask the WM for information about people? 3C12B

What does the WM return in response to an inquiry. 3C12C

Who maintains the data base? i.e. who can enter, and validate the information in the file. We see this as a big, on-going problem area. 3C12D

KIRK 4-FEB-75 01:15 25290

Jump to name External bug

Typing ctrl-o in jump to name external can cause it to cease working and print "fst entry nonexistant" from then on. Has been rumered to cause bad files. 1

KIRK 4-FEB-75 01:15 25290

Jump to name External bug

(J25290) 4-FEB-75 01:15;;;; Title: Author(s): Kirk E. Kelley/KIRK; Distribution: /FEED([ACTION]) BUGS([INFO-ONLY]) JDH([INFO-ONLY]) ; Sub-Collections: SRI-ARC BUGS; Clerk: KIRK;

JAKE 4-FEB-75 02:01 25291

Kjell Samuelson visit

Kjell called to say he will be here on Thurs. arriving about 11 o'clock. I believe he may have some other commitments later in the day so thought we could have lunch at I-bldg and then interact with him from 1 to 3. Let me know if you can make it for lunch and if you want to talk to him later. Jake 1

JAKE 4-FEB-75 02:01 25291

Kjell Samuelson visit

(J25291) 4-FEB-75 02:01;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /DCE([ACTION]) JHB([ACTION]) RLL([ACTION]) RA3Y([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

POOH 4-FEB-75 10:11 25292

<CTRL-O>

It appears that <CTRL-O> doesn't work when you are using the command Playback Record. When yo are playing back your recorded session and use <CTRL-O>, all sorts of different weird things begin to happen, ie.the syntax for all the commands in the base subsystem started to show on the screen. 1

<CTRL-O>

(J25292) 4-FEB-75 10:11;;;; Title: Author(s): Ann Weinberg/POOH; Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: POOH;

CHI 4-FEB-75 11:24 25293

tNice work on debugger memo 25286

Ken, ccongratulations on the debugger memo. Very nicely done! Dick will have to comment on the commitments made or implied by the memo before it goes to NSW people, but I think it should go out soon to let them know what we are thinking. -- Charles. 1

CHI 4-FEB-75 11:24 25293

tNice work on debugger memo 25286

(J25293) 4-FEB-75 11:24;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /NPG([INFO-ONLY]) RWW([INFO-ONLY]) ;
Sub-Collections: SRI-ARC NPG; Clerk: CHI;

RLL 4-FEB-75 16:17 25296

Sug:New Command to generate complete link from BUG

I think a new command should be implemented that inserts a link in a file from the effective designated address. 1

Most often it would take a BUG and generate a link with proper directory, filename, and statement numbers. Options will be possible. This command could be also useful when specifying a particular address, which is a complex of several statement numbers and/or structural and/or text relationships (e.g., 2a1.s.1.3p+5w). In this sense the link is generated from the effective address. 2

Suitable subcommand options should allow for no directory, no filename, no address statement identifier) (STID), character position, or viewspecs. 3

The default elements will be no directory, filename, statement number, and no character position. 4

```
"GENERATE" "LINK" <"from"> SOURCE <"with defaults">                    4a

   ("ALL" <"elements"> % include directory as well as all other
   defualts including character position%                              4a1

   / "YES" %yes take defaults %                                        4a2

   / CA % same as YES %                                                4a3

   / "NO"  %take the defaults with these modifications %               4a4

      REPEAT                                                           4a4a

         (<"option">     %or whatever mechanism to repeat this
         request for option%                                          4a4a1

         "DIRECTORY"                                                  4a4a2

         /"NOFILE" <"name">                                           4a4a3

         /"STATEMENT"                                                 4a4a4

            ("NULL"!L2!                                             4a4a4a

            / "STID"                                                4a4a4b

            / "NAME"!L1!)                                           4a4a4c

         /"CHARACTER" <"position">                                    4a4a5

         / CA                                                         4a4a6
```

Sug:New Command to generate complete link from BUG

/"FINISHED") %this repeats until command FINISHED or CA is specified % 4a4a7

) 4a4a8

VIEWSPEC 4b

Examples: 5

(1) Generate Link "from" BUG "with defaults" CA wg CA 5a

GENERATED LINK = <testfile,3a1;wg> 5a1

(2) Generate Link "from" BUG "with defaults" ALL CA 5b

GENERATED LINK = <LIEBERMAN,testfile,3a1> 5b1

(3) Generate link "from" BUG "with defaults" No "option" Nofile "option" Statement Stid "option" CHARACTER "position" "option" Finished OK 5c

GENERATED LINK = <0231+13c) 5c1

If the directory option given then filename will be generated (NOFILE name option will be ignored). 6

If the statement NULL option is given, then the character position option will be ignored. 7

RLL 4-FEB-75 16:17 25296

Sug:New Command to generate complete link from BUG

(J25296) 4-FEB-75 16:17;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /FEED([ACTION]) ARC-APP([INFO-ONLY]) ; Sub-Collections: SRI-ARC ARC-APP; Clerk: RLL;

RLL 4-FEB-75 16:36 25297

Visit: McMann of Standards and Poor 23 Jan 75

this is a contact report of a demo given.

Visit: McMann of Standards and Poor 23 Jan 75

(DATE) 23 Jan 75 1

(BY) Lieberman 2

(ATTENDEES) 3

Tom McMann of S&P 3a

Jim --- of SRI Financial Planning & Management Dept. 3b

Lieberman (RLL) of SRI 3c

(MEDIUM) FACE-TO-FACE 4

(WHERE) SRI-ARC, Menlo Park, CA 5

(ACTION-ITEMS) 6

None 6a

(DISTRIBUTION) JCN DCE RLL 7

(REFERENCES) 8

(REMARKS) 9

We received a phone call from Spetzler's office asking if it would be all right to give a demonstration to a visitor from Standards and Poor. 9a

The Financial Planning and Management (FPRM) Department of SRI has a contract from Standards and Poors (S&P), a large financial and business information consulting and service company. Jim of FP&M brought Tom McMann of S&P for the demo. 9b

I found Tom to be attentive and curious. He was well aware what facilities such as NLS could do for an organization like his. On leaving, he seemed that he might be in touch with us, but I am not sure this was not out of politeness. 9c

As for Jim of SRI, I felt he did not have any particular interest in our concept or facilities. 9d

(ADDRESSES) Full name of organization, address, and phone number 10

Tom McMann 10a

Standards and Poor 10a1

Visit: McMann of Standards and Poor 23 Jan 75

(DOCUMENTS) Hard copy given and received 11

(GIVEN) Date and documents given 11a

Coordinated Information Services for a Discipline- or Mission-Oriented community (mjournal,12445,) 11a1

The Augmented Knowledge Workshop, (ijournal,14724,) 11a2

(RECEIVED) Date and documents received 11b

RLL 4-FEB-75 16:36 25297

Visit: McMann of Standards and Poor 23 Jan 75

(J25297) 4-FEB-75 16:36;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /DCE([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

EKM DVN 4-FEB-75 17:09 25298

Promissing Application of NLS to Documentation Needs more Planning NOW to Forstall Serious Problems

Betty Finney who works with Elizabeth Riddle called today to 'check out the last details' before going into production on AFM 66-1. I had no idea what this was all about so she told me the following: 1

The Air Force Data Center has decided definitely to use NLS on an on-going production and maintenance basis to publish AFM 66-1. This is a 4000-page manual containing approximately 25,000,000 characters. 1a

The AFDC has agreed to purchase a full Office-1 slot plus an additional $15,000 of disk storage beginning July 1, 1975. Until that date the AFM 66-1 project will have the use of small amounts of the slots ARPA has allocated to the Air Force. 1b

Between now and July their energies will be devoted to getting the 4000 pages (now only in hardcopy) into machine-readable form. Beginnning next week, one (or more) typists will record the manual on IBM MTST cartridges. The MTST cartridges will be copied to IBM 360 tape using a Digi Data converter. This tape will be mailed to us (it may go through an intermediate IBM360 converion) to be converted to NLS files. 1c

Betty is in the process of making up formatting rules for the typists and is not completely clear on what these should be. I couldn't help much 1d

It is her (their?) understanding that by July 1 software will be available at Office-1 to structure the file, automatically insert statement lables, identify and process paragraph headings, insert output processor directives and produce the manual on the COM device of the AFDC's choice. 1e

At this point my courage failed. I said I would call back first thing tomorrow, said goodbye and hung up quickly before she could tell me how happy she is with graphics or speech strings or whatever. 1f

Dirk and I discussed the project and the potential problems. We agree that this appears to be a very good pilot project for a small to medium size document publication on a production basis. We also agree that without more project design, coordination and planning right now the project will be unnecessarily costly and probably fail. 2

We suggest strongly that Betty Finney delay the typist input work until the week beginning February 17th and that in the meantime the AFDC send us a hard copy of the documentation showing how headings should appear, tabs, indentation, etc, and exact specifcations of the translation steps the text will suffer between the MTST and the tape

we receive so we can issue an exact set of insttructions for the typists that will give us some hope of a smooth transition to NLS. 3

We are also concerned about their expectations for NLS editor, sequential file processing, automatic directives, and compatibility with a wide range of unknown COM devices particularly at Office-1. 4

We feel it is very important to get an understanding right now. 5

promissing Application of NLS to Documentation Needs more planning NOW to Forstall Serious Problems

(J25298) 4-FEB-75 17:09;;;; Title: Author(s): Elizabeth K. Michael, Dirk H. Van Nouhuys/EKM DVN; Distribution: /JOAN([ACTION] dpcs notebook please) WEC([ACTION]) LAC([ACTION]) RWW([ACTION]) RLL([ACTION]) JCN([ACTION]) DCE([ACTION]) JMB([INFO-ONLY]) EAR([INFO-ONLY]) VGK([INFO-ONLY]) ; Sub-Collections: DPCS SRI-ARC; Clerk: EKM; Origin: < MICHAEL, AFM66,NLS;1, >, 4-FEB-75 15:13 EKM ;;;;####;

)oug documentation.

i. The control characters that are available when in conference mode
ire:
- control W will tell you who has the floor.
- control E will take you to the exec, but keep DOUG in an
inferior exec you must type "QUIT" to return to DOUG.
- control U will give you the floor. To free the floor, just
finish you line with a CR, otherwise just keep typing.
- control D will allow the chairman (the 1st person in the
conference) to interupt the current speaker.
- control Z will exit the user from conference mode, and return
)im to DOUGS command level '*'.
- control N will enter you into NLS, just like the N command at
)OUGS command level.

1

)oug documentation.

(J25299) 4-FEB-75 22:05;;;; Title: Author(s): Geoffrey S.
;oodfellow/GSG; Distribution: /JAKE([INFO-ONLY]) ; Sub-Collections:
!IC; Clerk: GSG; Origin: < GEOFF, DOUG-HELP.NLS;3, >, 4-FEB-75
!2:01 GSG ;;;;####;

MLK 4-FEB-75 22:58 25300

IDENT ADDITIONS AND CHANGES

LARRY
I PUT THOSE IDENTS INTO THE IDENTFILE. ALSO MADE THE CORRECTIONS ON AAB, AJM AND YOUR IDENT THAT YOU REQUESTED. YOU MIGHT WANT TO CHECK ALL OF THEM OUT TO MAKE SURE EVERYTHING IS AS YOU WANT IT.
MARCIA

1

IDENT ADDITIONS AND CHANGES

(J25300) 4-FEB-75 22:58; Title: Author(s): Marcia Lynn Keeney/MLK;
Distribution: /FEED LAC; sub-Collections: SRI-ARC; Clerk: MLK;

test of unrecorded

what's it all about? 1

POOH 5-FEB-75 09:22 25301

test of unrecorded

(J25301) 5-FEB-75 09:22;;;; Title: (Unrecorded) Title: Author(s):
Ann Weinberg/POOH; Distribution: /KIRK([INFO-ONLY]) ;
Sub-Collections: SRI-ARC; Clerk: POOH;

POOH 5-FEB-75 09:57 25302

Undelete Modifications

Hi again!!! I am interested in the status of this command. It does not work at this point and the user gets the message not implemented. If it is not to be implemented, then I don't think the user should see it at all and it should be taken ot of all documentation. As it is now, documentation says that is does exist and describes how it is suppposed to work. In general, I have tried not to say things don't work, but rather describe how they should and then assume it is a bug. You may want to talk to me about this. 1

Undelete Modifications

(J25302) 5-FEB-75 09:57;;;; Title: Author(s): Ann Weinberg/POOH; Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: POOH;

DVN 5-FEB-75 11:50 25303

Micro-Datamation, A Possibility of Not-So-Fancy, Inexpensive COM

This morning Elizabeth Michael and I met with Gerald E. Roth, president of Micro-Datamation Corporation, 7806 Capwell Drive, Oakland, California 94621. Micro-Datamation is a COM supplier using a Bet-Gould 700. 1

With the Beta-Gould Micro-Datamation can make 48X reduced fiche (and indeed a range of reductions up to 84X). They have only one, sans cerif, type face but offer it in boldface or Italics, and can change size and underline. It comes either mono or proportionally spaced. They can layout the pages on the fiche in various ways and can extract titles. They can introduce slides (e.g.containing the lines of forms) between the video tube and the camera. 2

Roth was confident that if they could handle our output at all, they could produce masters at less than $5.00 not counting illustrations. 3

We gave Roth a copy of our output specifications (IJournal,14093,) and plan to make a tape for him today of a file formatted to Air Force Manual specifications. 4

DVN 5-FEB-75 11:50 25303

Micro-Datamation, A Possibility of Not-So-Fancy, Inexpensive COM

(J25303) 5-FEB-75 11:50;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION] dpcs notebook please) DPCS([INFO-ONLY]) VGK([INFO-ONLY]) FGB([INFO-ONLY]) WEC([INFO-ONLY]) LAC([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN; Origin: < HAMILTON, DATA-MATIONMTG,NLS;2, >, 5-FEB-75 10:56 JOAN ;;;;####;

DVN 5-FEB-75 16:53 25305

New NSW Documentation Work Loading and Preliminary Schedule

This document updates 24848 following conversations with RWW, EKM, POOH and KIRK and may help to explain 25158. To spread out work over the next few moths it is necessary to start documentation soon on the NSW tools that are most advanced even though changes in the tools may mean some rewriting later. Figures in square brackets indicate the number of full-time person weeks we expect to spend on the piece of work. Idents indicate assignments. 1

HELP(S)
By helps I mean covering a given subject in a help format, whether as a separate file or disperesed in a larger file. 2

NLS's New File structure. [1wk] KIRK Draft should be ready about April 21. 2a

This will involve scattered changes in the help information about files and about other matters affected by file's new properties. 2a1

COBOL Interface [2wks] KIRK Draft should be ready abou April 15 2b

Graphics [3wks] POOH, Starting 1/30, Draft should be ready 2/28 2c

DPCS [4wks] DVN Draft should be ready about March 1 2d

Includes getting the present system in and figuring out how to do that. 2d1

Sequential I/O [2wks] POOH Draft should be ready about May 1 2e

Mail I/O [2wk] POOH Draft should be ready Feb 21. 2f

NLS for "Inexperienced Users" [3wks] KIRK. He has started already started. Draft should be ready Feb 21. 2g

(Needs to be very good.) 2g1

Works Manager [.5wks] KIRK . 2h

All we need to do here is write specifications for a Help Data Base so MCA can follow them. 2h1

Miscellaneous Other Tools [4 wks] 2i

Command Summaries 3

COBOL Interface [.5wk] 3a

Graphics [.5wk] POOH 3b

New NSW Documentation Work Loading and Preliminary Schedule

DPCS [.5wk] DvN 3c

-Possibly including Official User Programs. 3c1

NLS "For Inexperienced Users" [.5wk] KIRK 3d

Mail I/O [.5 wks] POOH 3e

Primers 4

COBOL Interface [2wks] KIRK Draft Due April 30th 4a

Graphics [2wks] POOH Starrting 1/30 Draft should be ready 3/14. 4b

NLS "For Inexperienced Users" [2wks] KIRK Draft Due April 17th 4c

May be a re-write of the existing TNLS-8 Primer, in which case will take less time. 4c1

Mail I/O [1 wks] POOH Draft Due April 30th. 4d

Miscellaneous Other Tools [4 wks] 4e

Discursive Intrductions 5

COBOL Interface [1wk] KIRK Draft Due May 7. 5a

Graphics [1wk] POOH Starting 1/30,Draft due 3/21. 5b

DPCS [3wks] DVN Draft Due March 15 5c

A combination of documents explaining new features to old uers and of training materials for totally new users. 5c1

Mail I/O [1wk] POOH Draft due 3/14 5d

NLS for "Inexperienced Users" [1wks] KIRK Draft due 3/14 5e

(Needs to be very good, may be rewrite of Introduction to NLS.)) 5e1

Miscellaneous Other Tools [2 wks] 5f

Scenarios
(other than primers, may be more than one to a subject; here is where we would give ground first on priority.) 6

COBOL Interface [1wk]Kirk Draft Due May 15th 6a

New NSW Documentation Work Loading and Preliminary Schedule

DPCS [2Wk] DVN Draft Due May 1 6b

New features only + new introductory material 6b1

Graphics [1wk] POOH Start 1/30, Draft due 3/21, 6c

Sequential I/O [1wk] POOH Draft Due May 15 6d

Mail I/O [1wk] POOH, Draft due 3/14 6e

NLS for "Inexperienced Users" [1wks] KIRK Draft due 3/14 6f

Miscellaneous Other Tools [2 wks] 6g

Total person weeks: 52 7

The Labour Pool From February till July 8

POOH: One more week in February taken up by finishing Glossary, Leaves 19 weeks free, 8a

DvN: Will be spending a time decreasing irregularly from 75% to 25% on NSW documentaton. Planning, review, and Special projects will take up some of that time that time,Leaves 5 weeks free 8b

KIRK, Working Half Time on Documentation including reprogramming, A week in February takn up with reprogramming and running Glossary stuff, Leaves 8 weeks 8c

XXX who will be hired, Presumably can work close to full time on these projects, It will be the end of February before she can be useful, Leaves 16 weeks, We are depending on this new person, 8d

Total:51 weeks, That's cutting it pretty fine if there are schedule problems, 8e

Responsibilities and total weeks: 9

The thoughtful reader will see that more weeks have been aollocated to people in the subj ect outline than they posses in the Labor pool, That is becaue although XXX will take up some of the work load, all subjects except miscellaneous have been allocated and given due dates so spade work will begin and some one will be responible, 9a

Reporting 10

I will, and I ask POOH and KIRK please to copy their list of

responisbilities into their branch in the informal weekly report and comment on what they have done as it happens. 10a

DVN 5-FEB-75 16:53 25305

New NSW Documentation Work Loading and Preliminary Schedule

(J25305) 5-FEB-75 16:53;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /POOH([ACTION]) KIRK([ACTION]) JOAN([ACTION] dirt notebook please) DIRT([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT; Clerk: DVN; Origin: < VANNOUHUYS, NEWBREAK.NLS;2, >, 5-FEB-75 16:09 DVN ;;;;####;

test

test

DSM 7-FEB-75 13:32 25306

test

(J25306) 7-FEB-75 13:32;;;; Title: Author(s): David S. Maynard/DSM; Distribution: /DSM([ACTION]) DSM([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DSM;

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

INTRODUCTION 1

We have completed the first version of a document describing the NLS facilities available via PCP in the NSW. 1a

Some time ago you received from us a list of NLS core routines. You should check the new list of PCP-callable routines against the list of core routines to see if any routines you need have been left out of the list of PCP-callable routines. In the interests of making the NLS Backend tools easy to maintain, we've documented essentially the present command level routines as PCP-callable, but we don't consider this list cast in concrete by any means and are willing to make any required core routines PCP-callable if you need them. 1b

We are still considering some of the issues of argument conversion, and may decide to simplify some of the arguments to these routines. As we decide just what to do in this area, we'll keep you posted on any changes in the PCP-callable interface. 1c

You'll notice that a number of routines are preceded by the line:
"(ISI) Do you need ...."
These are primarily facilities that are either inappropriate in the NSW (e.g., TENEX directory manipulation) or are being handled by the Works Manager (e.g., copying files). We haven't deleted any of these routines from the description in case you need them or something like them. However, we would prefer that they not be included in the documented interface to the NLS Backend if you don't need them. 1d

We should get together in the next week or two to discuss some of these areas. Can some of you come to SRI sometime soon? 1e

NLS FRONTEND-BACKEND INTERFACE 2

The following (Journal document # 25304) is some brief documentation of the current design for the interface between the NSW Front-End Machine and the NLS Back-End process. This document defines the PCP encoding of the arguments passed to the NLS Back End. This specifies how one drives NLS through PCP and can be used by designers of other processes which wish to use the NLS Back-End process. The sections entitled "PRELIMINARY DEFINITIONS" and "PCP ARGUMENT ENCODINGS" will be of most use to these people. 2a

A partial list of the associated documentation follows: 2b

(24459,) "The Procedure Call Protocol" 2b1

(24460,) "The Procedure Interface Package" 2b2

(24461,) "The PCP Support Package" 2b3

(24462,) "The Process Management Package" 2b4

(24576,) "PCP Data Structure Formats" 2b5

(24792,) "Tenex PCP Process Internal Structure" 2b6

(20438,) "CML Documentation" 2b7

(25056,) "New CML Features for NSW" 2b8

PRELIMINARY DEFINITIONS: 2c

The following shorthands are used to denote the data structures which the built-in front end parse functions builds to specify character positions pointed at by a user. 2c1

A PSEL* specifies a selected character position on the screen of a display terminal. 2c2

PSEL* ==> LIST(%window-id% INTEGER, %string-id% INTEGER , %character-count% INTEGER) 2c2a

A TSEL* is a data structure which specifies a two pointing selections by the user of a display terminal. 2c3

TSEL* ==> LIST(PSEL* , PSEL*) 2c3a

An ANYSEL* is either a PSEL* or a TSEL* 2c4

ANYSEL* ==> PSEL* / TSEL* 2c4a

An ASELECTOR* is a data structure which specifies an entity selection in terms of the entity type and the "address" where the entity is to be found. An address is simply a CHARSTR which a back end process interprets to specify a location within a file. 2c5

ASELECTOR* ==> LIST(%entity-type% INTEGER, %mode% 2c5a

INTEGER[ADDRESS=1], %address%CHARSTR) 2c5a1

A PSELECTOR* is a data structure which specifies an entity selection in terms of an entity type and a selected character position on a display screen. 2c6

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

PSELECTOR* ==> LIST(%entity-type% INTEGER, %mode% 2c6a

INTEGER[POINT=0], %location% ANYSEL*) 2c6a1

FRONT END BUILTIN SELECTION FUNCTIONS 2d

Selection functions 2d1

Collect a literal selection from the user 2d1a

LSEL(entity-type => selection) 2d1a1

This procedure interacts with the user to get him to supply a literal of a given type. The user may specify this by either typing in a literal, by pointing to an entity on the screen, or by supplying an address where the literal can be found (addresses are only meaningful to the Back-end process). 2d1a2

result types 2d1a3

The format of the data structure returned by this parse function depends on how the user made the selection. 2d1a3a

If the user types a literal or points to an entity : 2d1a3b

selection - CHARSTR 2d1a3b1

If the user specifies an address at which to find the entity: 2d1a3c

selection - ASELECTOR* 2d1a3c1

Collect a source selection from the user 2d1b

SSEL(entity-type => selection) 2d1b1

This procedure interacts with the user to get him to supply a source selection of a given type. The user may specify this by either typing in a source selection, by pointing to an entity on the screen, or by supplying an address where the source selection can be found (addresses are only meaningful to the Back-end process). 2d1b2

result types 2d1b3

The format of the data structure returned by this

parse function depends on how the user made the selection. 2d1b3a

If the user types a literal: 2d1b3b

selection - CHARSTR 2d1b3b1

If the user points to the entity: 2d1b3c

selection - PSELECTOR* 2d1b3c1

If the user specifies an address at which to find the entity: 2d1b3d

selection - ASELECTOR* 2d1b3d1

Collect a destination selection from the user 2d1c

DSEL(entity-type => selection) 2d1c1

This procedure interacts with the user to get him to supply a destination selection of a given type. The user may specify this by either pointing to an entity on the screen, or by supplying an address where the destination selection can be found (addresses are only meaningful to the Back-end process). 2d1c2

result types 2d1c3

The format of the data structure returned by this parse function depends on how the user made the selection. 2d1c3a

If the user points to an entity: 2d1c3b

selection - PSELECTOR* 2d1c3b1

If the user specifies an address at which to find the entity: 2d1c3c

selection - ASELECTOR* 2d1c3c1

Builtin Entity Types: 2d2

The front end will support selections of the following entity types: 2d2a

TEXT , CHARACTER , WORD , VISIBLE , STRING , NEWFILENAME , OLDFILENAME , INTEGER , REAL , PASSWORD , INVISIBLE 2d2a1

See Appendix I for the association of integers and entity types assumed by the NLS Back End. 2d2b

NLS PARSE FUNCTIONS: 2e

The following parse functions will be supplied for selections of NLS entities which are not supported by the Front-End Builtin selection functions. 2e1

viewspecs 2e1a

This parse function will collect viewspec characters from the user and compute two an updated viewspec record. 2e1a1

VIEWSPECS(=> vwspint) 2e1a2

result type: 2e1a3

vwspint = %updatedvsrecord% BITSTR (72 bits) 2e1a3a

levadj 2e1b

This parse function will collect level adjust characters from the user and compute a level count. 2e1b1

LEVADJ(=> levadjint) 2e1b2

result type: 2e1b3

levadjint = INTEGER 2e1b3a

getabug 2e1c

This parse function is designed to be used as the pointing selection function for entities which require one bug from the user. 2e1c1

GETABUG(entity-type => result) 2e1c2

Argument/result type: 2e1c3

entity-type = INTEGER 2e1c3a

result = LIST(%entity-type% INTEGER, %mode% INTEGER 2e1c3b

[POINT=0], PSEL*) 2e1c3b1

gtwobugs 2e1d

This parse function is designed to be used as the pointing selection function for entities which require two bugs from the user. 2e1d1

GETABUG(entity-type -> result) 2e1d2

Argument/result type: 2e1d3

entity-type - INTEGER 2e1d3a

result - LIST(%entity-type% INTEGER, %mode% INTEGER 2e1d3b

[POINT=0], TSEL*) 2e1d3b1

Note that the data structures returned by getabug and gtwobugs are both legal examples of a PSELECTOR*. 2e1e

PCP ARGUMENT ENCODINGS 2f

The following section summarizes the encoding of arguments which are passed to various NLS packages via PCP. The encodings given here are those used by the NSW Front-End and NLS Back-End packages. The NLS packages can however be driven through PCP by any arbitrary process by making the proper PCP procedure calls and using the following PCP argument encodings. 2f1

VWSPEXT* - %viewspec collection string % CHARSTR 2f1a

VWSPINT* - %updatedvsrecord% BITSTR (72 bits) 2f1b

VIEWSPEC* - VWSPINT* / VWSPEXT* 2f1c

LEVADJEXT* - %level adjust collection string% CHARSTR 2f1d

LEVADJINT* - %relative level count% INTEGER 2f1e

LEVADJ* - LEVADJINT* / LEVADJEXT* 2f1f

LSEL* - selection 2f1g

Argument types 2f1g1

selection- one of the following 2f1g1a

CHARSTR 2f1g1a1

[implies that either the user typed a literal or pointed to a entity type known to the Front End] 2f1g1a1a

PSELECTOR* 2f1g1a2

[implies that the user pointed to an entity type unknown to the Front End] 2f1g1a2a

ASELECTOR* 2f1g1a3

[implies the user specified an address expression at which the NLS BE will locate the desired entity] 2f1g1a3a

see Appendix I for a list of the entity types . 2f1g1b

SSEL* - selection 2f1h

Argument types 2f1h1

selection- one of the following 2f1h1a

CHARSTR 2f1h1a1

[implies that the user typed a literal] 2f1h1a1a

PSELECTOR* 2f1h1a2

[implies that the user pointed to the entity] 2f1h1a2a

ASELECTOR* 2f1h1a3

[implies the user specified an address expression at which the NLS BE will locate the desired entity] 2f1h1a3a

DSEL* - selection 2f1i

Argument types 2f1i1

selection- one of the following 2f1i1a

PSELECTOR* 2f1i1a1

[implies that the user pointed to the entity] 2f1i1a1a

ASELECTOR* 2f111a2

[implies the user specified an address expression at which the NLS BE will locate the desired entity] 2f111a2a

PCP ARGUMENT DECODING 2g

The NLS BE packages will contain procedures which decode the PCP arguments and if possible check the validity of the arguments. These conversion routines will be called from the procedure dispatcher for each package. The dispatcher will perform temporary "Help" type returns upon encountering invalid arguments. It is possible that the NLS procedures will have to do further decoding of the arguments. The conversion/decoding procedures will therefore be available to the NLS procedures. 2g1

APPENDIX I ENTITY TYPES 2h

NSW ENTITY TYPES - 2h1

The following entity types are supported by the NSW Front End Machine. The Front End builtin function LSEL given one of the following entity types and a character position pointed at by the user will return a character string. 2h1a

2h1b

ENTITY	Entity type	
		2h1d
TEXT	1	2h1e
CHARACTER	2	2h1f
WORD	3	2h1g
VISIBLE	4	2h1h
STRING	5	2h1i
NEWFILENAME	6	2h1j
OLDFILENAME	7	2h1k
INTEGER	8	2h1l

REAL	9	2h1m
PASSWORD	10	2h1n
INVISIBLE	11	2h1o

NLS ENTITY TYPES - 2h2

The NLS Back End supports all of the NSW entity types, in addition it supports the following entities as arguments to the Front End selection routines. 2h2a

2h2b

ENTITY	Entity type	
		2h2d
BRANCH	26	2h2e
GROUP	27	2h2f
PLEX	28	2h2g
STATEMENT	29	2h2h
LINK	30	2h2i
DIRECTORY	31	2h2j
NAME	32	2h2k
EDGE	33	2h2l

NLS EDITOR 3

DEFINITIONS 3a

STRUCENT* := INTEGER[BRANCH=26/GROUP=27/PLEX=28/STATEMENT=29] 3a1

TEXTENTITY* := INTEGER[CHARACTER=2/WORD= 3/VISIBLE= 4/
INVISIBLE=11/TEXT= 1/LINK=30/INTEGER= 8/REAL=
9] 3a2

PROCEDURE DESCRIPTIONS 3b

Record user interaction 3b1

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

RECORDSESSION (desttype, destination, output)
PCALL RECORDSESSION (output) 3b1a

This co-routine adds the character string OUTPUT to the DESTINATION (of type DESTTYPE). Following the initial call, the Frontend repeatedly performs co-routine calls (PCALLs) to this routine as a result of the construct OUTPUT TO being encountered in the tool's grammar. When no more character strings are to be placed in DESTINATION, OUTPUT has the value EMPTY causing a normal termination return rather than a co-routine return. 3b1b

This co-routine is called to perform the NLS commands to construct a record of a session and to stop recording. 3b1c

Argument/result types: 3b1d

desttype - INTEGER[FILE=51]/STRUCENT* 3b1d1

destination - LSEL*(NEWFILENAME)/DSEL*(STRUCENT) 3b1d2

output - CHARSTR/EMPTY 3b1d3

Get user input from specified source 3b2

GETINPUT (sourcetype, source, count => input)
PCALL GETINPUT (count => input) 3b2a

This co-routine returns a buffer of characters (COUNT characters long) from the "next place" in SOURCE (which is of type SOURCETYPE). Following the initial call, the Frontend repeatedly performs co-routine calls (PCALLs) to this routine as a result of the construct INPUT FROM being encountered in the tool's grammar. When input is no longer to be obtained from SOURCE, COUNT has the value zero causing a normal termination return rather than a co-routine return. INPUT is returned EMPTY if SOURCE contains no more text. 3b2b

This co-routine is called to perform the NLS commands to playback a record of a session, to stop the playback, or to process a list of commands. 3b2c

Argument/result types: 3b2d

sourcetype - INTEGER[FILE=51]/STRUCENT* 3b2d1

source - LSEL*(OLDFILENAME/STRUCENT*) 3b2d2

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

count - INTEGER 3b2d3

input - CHARSTR/EMPTY 3b2d4

Append statement 3b3

APPEND (source,destination,literal) 3b3a

This procedure adds the text of the statement,SOURCE, to the end of the statement, DESTINATION. The character string, LITERAL, is inserted between the text of the two statements. The statement, SOURCE, is deleted. 3b3b

The curmkr is set to point between the text, LITERAL, and the text of the statement, SOURCE. 3b3c

Argument/result types: 3b3d

source - SSEL*(STATEMENT) 3b3d1

destination - DSEL*(STATEMENT) 3b3d2

literal - LSEL*(TEXT) 3b3d3

(ISI) Do you need this facility?
Archive file 3b4

ARCHIVE (filename,parameters => message) 3b4a

This procedure modifies certain bits in the TENEX FDB that control the archive status of the file or files. 3b4b

It accepts parameters to delete after archiving, defer archive, prevent deletion after archiving, not allow archiving, and reset archive bits to zero. 3b4c

This procedure returns a list of character strings indicating those files whose archive status has been changed. 3b4d

Argument/result types: 3b4e

filename - LSEL*(OLDFILENAME) 3b4e1

parameters - LIST (INTEGER[DELETE=50]/EMPTY, INTEGER[DEFERRED=52]/EMPTY, INTEGER[NOT=53]/EMPTY, INTEGER[PREVENT=54]/EMPTY, INTEGER[RESET=55]/EMPTY) 3b4e2

message - LIST(CHARSTR) 3b4e3

Break Statement 3b5

BREAK (breakplace, level) 3b5a

This procedure breaks a statement into two statements. It breaks the statement after the second character position pointed to by BREAKPLACE. A LEVEL relative to the original statement may be specified for the new statement. 3b5b

The curmkr is set to the beginning of the new statement. 3b5c

Argument/result types: 3b5d

breakplace - DSEL*(TEXTENTITY*) 3b5d1

level - LEVADJ* 3b5d2

Compile 3b6

COMPILE (type, location, compiler, object) 3b6a

This procedure compiles the source code located at LOCATION, of type TYPE, using COMPILER. If TYPE has the value FILE, a file OBJECT is produced; otherwise, the compiled code is loaded into the process, either as a content analyzer program (for TYPE = CONTENT) or as a user program (for TYPE = L10). 3b6b

Argument/result types: 3b6c

type - INTEGER[FILE=51/L10=52/CONTENT=53] 3b6c1

FILE: location - DSEL*(STATEMENT)
compiler - LSEL*(OLDFILENAME)
object - LSEL*(NEWFILENAME) 3b6c1a

L10: location - DSEL*(STATEMENT)
compiler - EMPTY
object - EMPTY 3b6c1b

CONTENT: location - LSEL*(TEXT)
compiler - EMPTY
object - EMPTY 3b6c1c

(ISI) Do you need this facility?
Connect to NLS directory 3b7

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

CONNECT (dirname) 3b7a

This procedure connects the user to the NLS directory, DIRNAME. 3b7b

Argument/result types: 3b7c

dirname - LSEL*(directory) 3b7c1

(ISI) Do you need Copy File and/or Copy Directory?
Copy 3b8

COPY (sourcetype, source, desttype, destination, level, filter => copiedlist) 3b8a

This procedure copies source pointed to by SOURCE, of type SOURCETYPE, to a DESTINATION of type DESTTYPE. The source may be a text string, structure, an NLS directory, NLS files, or a sequential file. When files are copied, a list of the files copied (COPIEDLIST) is returned. 3b8b

When a structure (statement, branch, group, plex) is being copied a FILTER may be specified (if FILTERFLAG is TRUE). This is a viewspec specifying level and/or content analysis status. 3b8c

The curmkr is set to the beginning of the (first) statement copied to (or copied into). 3b8d

Argument/result types: 3b8e

sourcetype - TEXTENTITY*/STRUCENT*/
INTEGER[FILE=51/DIRECTORY=9/SEQUENTIAL=52]) 3b8e1

TEXTENTITY: source - SSEL*(sourcetype)
desttype - TEXTENTITY*
destination -
DSEL*(TEXTENTITY*)
level - EMPTY
filter - EMPTY
copiedlist - EMPTY 3b8e1a

STRUCENT: source - SSEL*(sourcetype)
desttype -
INTEGER[STATEMENT=29]
destination - DSEL*(STATEMENT)
level - LEVADJ*

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

```
                                filter - EMPTY/VIEWSPEC*
                                copiedlist - EMPTY                3b8e1b

        FILE:           source - LSEL*(OLDFILENAME)
                                desttype - INTEGER[FILE=51]
                                destination -
        LSEL*(NEWFILENAME)
                                level - EMPTY
                                filter - EMPTY
                                copiedlist - LIST(CHARSTR)        3b8e1c

        SEQUENTIAL:     source - LSEL*(OLDFILENAME)
                                desttype -
        EMPTY/INTEGER[TWO=53/
                                JUSTIFIED=54]
                                destination - DSEL*(STATEMENT)
                                level - LEVADJ*
                                filter - EMPTY
                                copiedlist - LIST(CHARSTR)        3b8e1d

        DIRECTORY:              source - LSEL*(DIRECTORY)
                                desttype - DIROPTIONS*
                                destination - DSEL*(STATEMENT)
                                level - LEVADJ*
                                filter -
        EMPTY/LSEL(OLDFILENAME)
                                copiedlist - EMPTY                3b8e1e

        DIROPTIONS* := LIST(
                        INTEGER[BOTH=81]/EMPTY,
                        INTEGER[DELETE=50]/EMPTY,
                        INTEGER[UNDELETE=82]/EMPTY,
                        INTEGER[FOR=83]/EMPTY,
                        LIST(INTEGER[ARCHIVE=61],
                        INTEGER[STATUS=52/TAPE=84]),
                        INTEGER[ACCOUNT=85]/EMPTY,
                        LIST(INTEGER[DATE=51]
                        ,INTEGER[ARCHIVE=61/CREATION=86/

        LAST=51/FIRST=87/READ=88/WRITE=89]),
                        INTEGER[DUMP=90]/EMPTY,
                        INTEGER[EVERYTHING=91]/EMPTY,
                        INTEGER[LAST=51]/EMPTY,
                        INTEGER[LENGTH=92]/EMPTY,
                        INTEGER[MISCELLANEOUS=93]/EMPTY,

        LIST(INTEGER[INTEGER=11],INTEGER[VERSIONS=94/
                                ACCESSES=95]),
```

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

```
                        LIST(INTEGER[NO=96],
                        INTEGER[VERSIONS=94/EXTERSION=97]),
                        INTEGER[PROTECT=98]/EMPTY,
                        INTEGER[SIZE=99]/EMPTY,

            LIST(INTEGER[TIME=52],INTEGER[ARCHIVE=61/

            CREATION=86/LAST=51/FIRST=87/READ=88/
                                WRITE=89]),
                        INTEGER[VERBOSE=100]/EMPTY,

            LIST(INTEGER[GROUp=2]<INTEGER[REVERSE=101]/EMPTY,

            INTEGER[NO=96/ACCOUNT=85/ARCHIVE=61/

            CREATION=86/DELETE=50/DUMp=90/FIRST=87/

            LAST=51/INTEGER=11/READ=88/WRITE=89]/EMPTY,

            INTEGER[DATE=51/STATUS=52/TAPE=84]/EMpTY)
                        )                                          3b8e1f
```

(ISI) Do you need this facility?
Create File 3b9

CREATE (filename, window => originadr) 3b9a

This procedure creates a file named FILENAME in WINDOW (if a window is specified; otherwise, the file is not associated with a window) and returns ORIGINADR, the index (into a table of statement addresses) of the internal NLS address of the origin statement of the file. 3b9b

The curmkr is set to the origin statement of the new file. 3b9c

Argument/result types: 3b9d

filename - LSEL*(NEWFILENAME) 3b9d1

window - BSEL*/EMPTY 3b9d2

originadr - INTEGER 3b9d3

(ISI) Do you need Delete File?
Delete 3b10

DELETE (enttype, entity, filter) 3b10a

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

This procedure deletes text entities, structural entities, a specified marker, or ALL markers from files. It also deletes files, display windows (EDGE), file modifications (partial copies), or files (the last one loaded or all of them) from the user programs buffer. 3b10b

For textual entities, the curmkr is set to the next such entity or (if it would be beyond the end of the statement) to the last character in the statement. For structural entities, the curmkr is set to the next such entity or (if it would be beyond the end of the file) to the entity before the one(s) deleted. For all other entities, the curmkr is unchanged. 3b10c

Argument/result types: 3b10d

enttype =
TEXTENTITY*/STRUCENT*/INTEGER[MARKER=51/PROGRAMS=52/
EDGE=21/MODIFICATIONS=53] 3b10d1

TEXTENTITY: entity - DSEL*(TEXTENTITY)
filter - EMPTY 3b10d1a

STRUCENT: entity - DSEL*(STRUCENT)
filter - VIEWSPEC* 3b10d1b

FILE: entity - LSEL*(OLDFILENAME)
filter - EMPTY 3b10d1c

MARKER: entity -
INTEGER[ALL=52]/LSEL*(MARKER)
filter - EMPTY 3b10d1d

PROGRAMS: entity -
INTEGER[LAST=51/ALL=52]
filter - EMPTY 3b10d1e

EDGE: entity - DSEL*(EDGE)
filter - EMPTY 3b10d1f

MODIFICATIONS: entity - EMPTY
filter - EMPTY 3b10d1g

Disestablish user program 3b11

DISESTABLISH (type) 3b11a

This procedure disestablishes the program of the type TYPE that is currently established. 3b11b

This procedure resets values in a system table. 3b11c

Argument types: 3b11d

type - INTEGER[CONTENT=53/SORT=52/SEQGEN=51] 3b11d1

Establish user program 3b12

ESTABLISH (type, filter) 3b12a

This procedure establishes the user program FILTER as the current content analyzer program, the current sort key extractor program, or the current sequence generator program depending on the value of TYPE. 3b12b

This procedure sets values in a system table. 3b12c

Argument types: 3b12d

type - INTEGER[CONTENT=53/SORT=52/SEQGEN=51] 3b12d1

filter - LSEL*(NAME) 3b12d2

(ISI) Do you need this facility?
Expunge a directory 3b13

EXPUNGE (directorytype) 3b13a

This procedure expunges either the connected or the archive directory depending on DIRECTORYTYPE. 3b13b

Argument/result types: 3b13c

directorytype - INTEGER[DIRECTORY=9/ARCHIVE=61] 3b13c1

Force case 3b14

FORCE (type, case, location) 3b14a

This procedure changes the case of text of type TYPE at LOCATION depending on the value of CASE (or a global case mode variable if CASE is EMPTY). It changes the value of the global case mode variable if TYPE has the value MODE. The four permissible values of CASE produce 1) all upper case, 2) all lower case, 3) first character of each word

upper case, all other characters lower case, or 4) first character of each sentence upper case. 3b14b

The curmkr is set to LOCATION. 3b14c

Argument/result types: 3b14d

type - TEXTENTITY*/STRUCENT*/INTEGER[MODE=51] 3b14d1

TEXTENTITY/STRUCENT: location - DSEL*(type) 3b14d1a

MODE: location - EMPTY 3b14d1b

case - EMPTY/INTEGER %1 = lower case, 2 = first char upper, 3 = upper case, 4 = sentence upper case% 3b14d2

Freeze statement 3b15

FREEZE (location, viewspecs) 3b15a

This procedure freezes the statement at LOCATION with the display form controlled by VIEWSPECS. Freezing a statement allows it to be displayed in a special portion of the window, where it remains (regardless of jumps within the window) until explicitly removed. 3b15b

Argument/result types: 3b15c

location - DSEL*(STATEMENT) 3b15c1

viewspecs - VIEWSPEC* 3b15c2

Insert 3b16

INSERT (type, destination, level, source) 3b16a

This procedure inserts the entity SOURCE of type TYPE to follow DESTINATION (following the second character position specified for a textual type or the statement specified for a structural type). A LEVEL relative to DESTINATION may be specified for a new structural entity or a sendmail form. The DATE and TIME types cause the TENEX date or time string to be inserted. 3b16b

The curmkr is set to the entity inserted when appropriate (i.e., for entity types other than DATE, TIME, and EDGE). 3b16c

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

Argument/result types: 3b16d

type= TEXTENTITY*/STRUCENT*/
INTEGER[DATE=51/TIME=52/SENDMAIL=53/EDGE=21] 3b16d1

TEXTENTITY: source = LSEL*(type)
destination = DSEL(TEXTENTITY)
level = EMPTY 3b16d1a

STRUCENT: source = LSEL*(type)
destination = DSEL*(STATEMENT)
level = LEVADJ* 3b16d1b

DATE/TIME: source = EMPTY
destination = DSEL*(TEXTENTITY)
level = EMPTY 3b16d1c

SENDMAIL: source = EMPTY
destination = DSEL*(STATEMENT)
level = LEVADJ* 3b16d1d

EDGE: source = EMPTY/INTEGER[CENTER=54]
destination = BSEL*(EDGE)
level = EMPTY 3b16d1e

Insert Statement 3b17

INSERTSTATEMENT (level, text) 3b17a

This procedure inserts TEXT as a statement in a file following the statement pointed to by the curmkr. A LEVEL may be specified for the new statement relative to the old statement. 3b17b

The curmkr is set to the newly inserted statement. 3b17c

Argument/result types: 3b17d

level = LEVADJ* 3b17d1

text = LSEL*(STATEMENT) 3b17d2

(ISI) Do you need Jump to Link, Jump to File, or Jump to File Named?

Jump 3b18

JUMP (type, location, filter, window) 3b18a

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

This procedure jumps to LOCATION (of type TYPE) with the view controlled by FILTER. The jump takes effect in the file displayed in the window, WINDOW. The effect of jumping is to change the current marker in the loaded file to be LOCATION or some statement relative to LOCATION, and (if the user's terminal is a display) to display a new portion of the file and possibly to change the user's view of the file. 3b18b

The curmkr is set to LOCATION or to the beginning of a statement relative to LOCATION (when a relative jump such as Jump to Successor is specified). 3b18c

Argument/result types: 3b18d

```
type =
KEY2*(STATEMENT=29/SUCCESSOR=52/PRECEDESSOR=53/UP=54/
DOWN=55/HEAD=56/TAIL=57/END=58/BACK=59/ORIGIN=60/NEXT=61/
LINK=8/RETURN=62/FILE=51/FILENAMED=63/NAME=18/FIRSTNAME=6
4/ NEXTNAME=65/EXTNAME=66/FILERETURN=67/FIRSTCONTENT=68/
NEXTCONTENT=69/FIRSTWORD=70/NEXTWORD=71/CHARACTER=5]      3b18d1

   STATEMENT/SUCCESSOR/PRECEDESSOR/UP/DOWN/HEAD/TAIL/END/
   BACK/ORIGIN/NEXT:                location -
   LSEL*(STATEMENT)
                                    filter - VIEWSPEC*    3b18d1a

   FILE:                    location - DSEL*(OLDFILENAME)
                                    filter - VIEWSPEC*    3b18d1b

   FILENAMED:                       location -
   LSEL*(OLDFILENAME)
                                    filter - VIEWSPEC*    3b18d1c

   CHARACTER:                       location -
   DSEL*(CHARACTER)
                                    filter - EMPTY        3b18d1d

   LINK:                    location - LSEL*(LINK)
                                    filter - EMPTY        3b18d1e

   NAME/FIRSTNAME/NEXTNAME/EXTNAME:
                                    location - LSEL*(NAME)
                                    filter - VIEWSPEC*    3b18d1f

   FIRSTCONTENT/NEXTCONTENT:        location -
   LSEL*(TEXT)/EMPTY
                                    filter - VIEWSPEC*    3b18d1g
```

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

FIRSTWORD/NEXTWORD: location - LSEL*(WORD)/EMPTY
filter - VIEWSPEC* 3b18d1h

RETURN/FILERETURN: location - INTEGER
filter - EMPTY 3b18d1i

window - BSEL*/EMPTY 3b18d2

Provide next statement or file return ring 3b19

RINGJUMP (type, index, window => result) 3b19a

This procedure returns the character string value of the next statement or file name (depending on TYPE) in the statement return ring or file return ring for the window, WINDOW. The INDEX can be passed to the Jump routine for the return or file return types (see above). 3b19b

Argument/result types: 3b19c

type - INTEGER[FILE=51/STATEMENT=29] 3b19c1

index - INTEGER 3b19c2

window - BSEL*/EMPTY 3b19c3

result - CHARSTR 3b19c4

(ISI) Do you need this facility?
Load File or program 3b20

LOAD (type, name, window => originadr) 3b20a

This procedure loads a user program into the user programs buffer or loads the file NAME into core. If a file is to be loaded, it gets a file and returns ORIGINADR, the index (into a table of statement addresses) of the internal NLS address of the origin statement of the file. The file is opened and the window, WINDOW, is updated for display terminals. If a program is to be loaded, the specified object file, NAME, is loaded into the next available location in the user programs buffer. 3b20b

The curmkr is set to the origin of the newly loaded file. 3b20c

When loading a user program, the actions taken depend upon the TENEX file extension of the object file. This may

correspond to the NSW file use type of the file. Currently the following actions are taken : 3b20d

Extension REL - 3b20d1

The object file is loaded into the users program buffer. 3b20d1a

Extension CA - 3b20d2

The object file is loaded into the users program buffer and established as a content analyser program. 3b20d2a

Extension SK - 3b20d3

The object file is loaded into the users program buffer and established as a sort key program. 3b20d3a

Extension SG - 3b20d4

The object file is loaded into the users program buffer and established as a sequence generator program. 3b20d4a

Extension PROC-REP - 3b20d5

The object file is loaded into the users program buffer. The first procedure in the file replaces any existing procedure of the same name in the process's address space. 3b20d5a

Extension SUBSYS - 3b20d6

The object file is loaded into the users program buffer and the procedures in the file are made into a new package within the process. This requires modifying the PCP tables. When the user subsequently goes to the tool the required package is available. 3b20d6a

Argument/result types: 3b20e

type - INTEGER[FILE=51/PROGRAM=52] 3b20e1

FILE: name - LSEL*(OLDFILENAME)
window - BSEL*/EMPTY
originadr - INTEGER 3b20e1a

PROGRAM: name - LSEL*(NAME)

window - EMPTY
originadr - EMPTY 3b20e1b

Mark 3b21

MARK (location, name) 3b21a

This procedure associates a NAME with a particular character, LOCATION, in a file. The name may be 1-5 alphanumeric characters includng hyphen and double quotes. 3b21b

The curmkr is set to the named character in the file. 3b21c

Argument/result types: 3b21d

location - DSEL*(CHARACTER) 3b21d1

name - LSEL*(NAME) 3b21d2

Merge 3b22

MERGE (source, destination) 3b22a

This procedure merges the set of presorted statements designated by SOURCE into the set of presorted statements designated by DESTINATION. 3b22b

The curmkr is set to the head of the merged set of statements. 3b22c

Argument/result types: 3b22d

source - SSEL*(GROUP/BRANCH/PLEX) 3b22d1

destination- DSEL*(GROUP/BRANCH/PLEX) 3b22d2

(ISI) Do you need Move File?
Move 3b23

MOVE (sourcetype, source, desttype, destination, level, filter -> movedlist) 3b23a

This procedure moves source pointed to by SOURCE, of type SOURCETYPE, to a destination of type DESTTYPE, located at DESTINATION and adjusted by LEVEL. The source may be a text string, structure, TENEX directory, NLS file, or sequential file. Only sources that pass FILTER are moved. When files

are moved, a MOVEDLIST is returned indicating the moved files. 3b23b

The curmkr is set to the moved entity in the new location for structural and textual entities. 3b23c

Argument/result types: 3b23d

sourcetype = TEXTENTITY*/STRUCENT*/INTEGER[FILE=51/EDGE=21] 3b23d1

TEXTENTITY: source = SSEL*(sourcetype)
desttype = same as sourcetype
destination = DSEL*(TEXTENTITY)
level = EMPTY
filter = EMPTY
movedlist = EMPTY 3b23d1a

STRUCENT: source = SSEL*(sourcetype)
desttype = INTEGER[STATEMENT=29]
destination = DSEL*(STATEMENT)
level = LEVAPJ*
filter = EMPTY/VIEWSPEC*
movedlist = EMPTY 3b23d1b

FILE: source = LSEL*(OLDFILENAME)
desttype = INTEGER[FILE=51]
destination = LSEL*(NEWFILENAME)
level = EMPTY
filter = EMPTY
movedlist = LIST(CHARSTR) 3b23d1c

EDGE: source = DSEL*(EDGE)
desttype = EMPTY/INTEGER[CENTER=54]
destination = DSEL*(EDGE)
level = EMPTY
filter = EMPTY
movedlist = EMPTY 3b23d1d

(ISI) Do you need this facility?
Output a formatted file 3b24

OUTPUT (type, destination, location, paramlist) 3b24a

This procedure formats and outputs as TYPE the loaded file

and puts it at DESTINATION. LOCATION is the statement from which the output is to start. PARAMLIST is a list of various parameters (and has different interpretations depending on TYPE). The first parameter is a network port number for output to a remote printer or a count of the number of copies to make for output to quickprint, journal, printer, or com. The second parameter equal TRUE indicates a test is being made (so an output com is printed locally instead of really being sent for composition) or that formfeeds are to be sent to the terminal or remote printer. The third parameter equal TRUE means that headers should be printed or that formfeeds should be simulated, depending on TYPE. The fourth parameter equal TRUE means that the loaded file should be appended to a sequential file or that the ouput should wait at page breaks, depending on TYPE. 3b24b

Argument/result types: 3b24c

```
type -
INTEGER[QUICKPRINT=51/JOURNAL=53/PRINTER=54/COM=55/
SEQUENTIAL=52/ASSEMBLER=56/TERMINAL=57/REMOTE=58]          3b24c1

   QUICKPRINT/JOURNAL/PRINTER/COM:
                  destination - EMPTY/LSEL*(OLDFILENAME/
                  NEWFILENAME)
                  location - SSEL*(STATEMENT)
                  paramlist - EMPTY/LSEL*(INTEGER)        3b24c1a

   SEQUENTIAL/ASSEMBLER:
                  destination -
   LSEL*(OLDFILENAME/NEWFILENAME)
                  location - EMPTY
                  paramlist - EMPTY                       3b24c1b

   TERMINAL:      destination - EMPTY/LSEL*(NEWFILENAME)
                  location - EMPTY
                  paramlist - LIST(BOOLEAN, BOOLEAN,
   BOOLEAN)                                               3b24c1c

   REMOTE:        destination - LSEL*(VISIBLE)
                  location - EMPTY
                  paramlist - LIST(BOOLEAN, BOOLEAN,
   BOOLEAN)                                               3b24c1d
```

Print 3b25

```
PRINT (type, source, filter => message)
PCALL PRINT (=> message)                                  3b25a
```

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

This co-routine returns the "next statement" of the specified part of the loaded file. The type of the part to be returned (statement by statement) is specified by TYPE and its location by SOURCE. Each statement of that part is returned or not, depending on whether or not it passes FILTER. Following the initial call, the Frontend repeatedly performs co-routine calls (PCALLs) to this routine to receive successive statements. When MESSAGE is returned EMPTY, the last statement of the specified part has been returned. 3b25b

ISI:
We think this routine could be modified to provide a virtual-text interface. We need to discuss the problem with you. 3b25c

The curmkr is not changed. 3b25d

Argument/result types: 3b25e

type - STRUCENT*/INTEGER[REST=52/FILE=51/JOURNAL=53] 3b25e1

STRUCENT: source - DSEL*(type)
filter - VIEWSPEC* 3b25e1a

REST/FILE/JOURNAL: source - EMPTY
filter - EMPTY 3b25e1b

message - CHARSTR/EMPTY 3b25e2

Print the next statement 3b26

PRINTNEXT (=> message) 3b26a

This procedure returns a character string containing the statement following the current one. 3b26b

The curmkr is set to the next statement. 3b26c

Argument/result types: 3b26d

message - CHARSTR 3b26d1

Print the previous statement 3b27

PRINTPREVIOUS (=> message) 3b27a

This procedure returns a character string containing the statement preceding the current one. 3b27b

The curmkr is set to the previous statement. 3b27c

Argument/result types: 3b27d

message - CHARSTR 3b27d1

Print the current statement location 3b28

PRINTSTMTNO (=> message) 3b28a

This procedure returns a character string containing the statement number of the current statement. 3b28b

The curmkr is not changed. 3b28c

Argument/result types: 3b28d

message - CHARSTR 3b28d1

Print the current context of the curmkr 3b29

PRINTCURCON (count -> message) 3b29a

This procedure returns a character string containing the COUNT characters before and after the location of the curmkr, marking the location of the curmkr. If COUNT is EMPTY the number of character specified in the user profile are returned. 3b29b

The curmkr is not changed. 3b29c

Argument/result types: 3b29d

count - LSEL*(INTEGER)/EMPTY 3b29d1

message - CHARSTR 3b29d2

Print the current statement 3b30

PRINTSTMT (=> message) 3b30a

This procedure returns a character string containing the current statement. 3b30b

The curmkr is not changed. 3b30c

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

Argument/result types: 3b30d

message - CHARSTR 3b30d1

Release frozen statements 3b31

RELEASE (type, location) 3b31a

This procedure releases TYPE (frozen) statements located at LOCATION. It either releases a particular frozen statement or all frozen statements. A frozen statement can be displayed in a special portion of the window, where it remains until explicitly removed regardless of jumps within the window. 3b31b

Argument/result types: 3b31c

type - INTEGER[FROZEN=51/ALL=52] 3b31c1

location - DSEL*(STATEMENT) 3b31c2

Renumber statement IDs 3b32

RENUMBER (destination) 3b32a

This procedure renumbers all statement IDs in the file specified by DESTINATION. 3b32b

The curmkr is not changed. 3b32c

Argument/result types: 3b32d

destination - DSEL*(CHARACTER)/LSEL*(OLDFILENAME) 3b32d1

Repeat the last search command 3b33

REPEATSEARCH () 3b33a

This procedure performs the previous search-type Jump command (Jump to Word Next, etc.) continuing from the current statement. 3b33b

The curmkr is set to the location jumped to. 3b33c

Replace 3b34

REPLACE (destination, source) 3b34a

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

The procedure replaces the NLS entity at DESTINATION by the entity at SOURCE. 3b34b

The current marker is set to the replaced entity. 3b34c

Argument/result types: 3b34d

destination - DSEL*(TEXTENTITY/STRUCENT) 3b34d1

source - LSEL*(TEXTENTITY/STRUCENT) 3b34d2

(ISI) Do you need Reset Archive status?
Reset the specified property 3b35

RESET (property, location) 3b35a

This procedure resets the item specified by PROPERTY. The reset takes effect at LOCATION, if appropriate. The properties that can be reset are archive status for a file, case mode, content analyzer pattern, link default (for directory names in links), name delimiters, temporary modifications for the loaded file, viewspecs, and the user programs buffer. 3b35b

Argument/result types: 3b35c

property - INTEGER[ARCHIVE=61/CASE=52/CONTENT=53/LINK=8/NAME=18/TEMPORARY=56/VIEWSPECS=57/BUFFER=58] 3b35c1

ARCHIVE: location - LSEL*(OLDFILENAME) 3b35c1a

NAME: location - DSEL*(STRUCENT) 3b35c1b

all the rest: location - EMPTY 3b35c1c

(ISI) Do you need this facility?
Retrieve a file from archive 3b36

RETRIEVE (filename) 3b36a

This procedure retrieves the file FILENAME from the archive where it is stored. 3b36b

Argument/result types: 3b36c

filename - LSEL*(OLDFILENAME) 3b36c1

Run user program 3b37

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

```
RUN (progname)                                                  3b37a

This procedure calls the user program PROGNAME and executes
it.                                                             3b37b

Argument types:                                                 3b37c

   progname - LSEL*(NAME)                                       3b37c1

(ISI)                   Do you need Set Tenex protection?
Set the specified property                                      3b38

SET (property, valuelist, location)                             3b38a

This procedure sets PROPERTY to the values in VALUELIST.
They take effect at LOCATION, if appropriate.  The
properties that can be set are content analyzer pattern,
external names link file (for Jump to Name External), link
default (for directory names in links), name delimiters, NLS
file protection (private or public), TENEX protection for a
file, viewspecs, and user programs buffer size.                 3b38b

Argument/result types:                                          3b38c

   property - INTEGER[CONTENT=53/EXTERNAL=59/LINK=8/NAME=18/
   PRIVATE=51/PUBLIC=54/TEMPORARY=56/TENEX=55/VIEWSPECS=57/
   BUFFER=58]                                                   3b38c1

      CONTENT:        valuelist -
      LIST(INTEGER[TO=61/ON=62/OFF=63],
                      LSEL*(CHARACTER)/EMPTY, EMPTY)
                      location - EMPTY                          3b38c1a

      EXTERNAL:       valuelist - LIST(LSEL*(LINK), EMPTY,
      EMPTY)
                      location - EMPTY                          3b38c1b

      LINK:  valuelist - LIST(EMPTY, LSEL*(NAME), EMPTY)
                      location - EMPTY                          3b38c1c

      NAME:  valuelist - LIST(STRUCENT*, LSEL*(CHARACTER),
                      LSEL*(CHARACTER))
                      location - DSEL*(value1)                  3b38c1d

      PRIVATE/PUBLIC/TEMPORARY:
                      valuelist - LIST(EMPTY, EMPTY, EMPTY)
                      location - EMPTY                          3b38c1e
```

TENEX: valuelist = LIST(LSEL*(OLDFILENAME), LIST(TENEXaccess))
location = EMPTY 3b38c1f

VIEWSPECS: valuelist = LIST(VIEWSPEC*, EMPTY, EMPTY)
location = EMPTY 3b38c1g

BUFFER: valuelist = LIST(LSEL*(INTEGER), EMPTY, EMPTY)
location = EMPTY 3b38c1h

Set/reset the current context length option 3b39

CURCON (count) 3b39a

This procedure sets the current context length to COUNT characters, or resets it to the default value if COUNT is EMPTY. The current context length is the number of characters to each side of the curmkr that are returned when the user requests that the current context of the marker be printed. 3b39b

Argument/result types: 3b39c

count = LSEL*(INTEGER)/EMPTY 3b39c1

Set/reset the external name file option 3b40

EXTNAMEFILE (filename) 3b40a

This procedure sets to FILENAME the name of the file to serve as an indirection list for connecting an external name with the file where it's found. The indirection list file is used during a Jump to Name External command. If FILENAME is EMPTY, the external name file is reset to its default value. 3b40b

Argument/result types: 3b40c

filename = LSEL*(LINK)/EMPTY 3b40c1

Set/reset the level adjust option 3b41

LEVELADJUST (switch) 3b41a

This procedure sets the level adjust flag in the tool's user options to indicate whether (LEVEL = TRUE) or not (LEVEL =

FALSE) level adjustment is to be done. If LEVEL is EMPTY, the level adjust flag is reset to its default value. 3b41b

Argument/result types: 3b41c

switch - BOOLEAN/EMPTY 3b41c1

Set/reset the name delimiter options 3b42

NAMEDELIM (left, right) 3b42a

This procedure sets the left and right name delimiters to the characters specified by LEFT and RIGHT. If either LEFT or RIGHT is EMPTY, that delimiter is reset to its default value. 3b42b

Argument/result types : 3b42c

left - LSEL*(CHARACTER)/EMPTY 3b42c1

right - LSEL*(CHARACTER)/EMPTY 3b42c2

Set/reset the print parameter options 3b43

PRINTPARAM (pparam, value) 3b43a

This procedure sets the user's print parameter PPARAM to VALUE. If VALUE is EMPTY, the specified parameter is reset to its default value. 3b43b

Argument/result types: 3b43c

pparam - INTEGER[RIGHT=51/LEFT=52/BOTTOM=53/PAGE=54/ INDENTING=55/TAB=56] 3b43c1

TAB: value - LSEL*(TEXT)/EMPTY 3b43c1a

all the rest: value - LSEL*(INTEGER)/EMPTY 3b43c1b

Set/reset the ring size option 3b44

RINGSIZE (ringtype, count) 3b44a

This procedure sets the RINGTYPE ring to contain a maximum of COUNT entries. If COUNT is EMPTY, the specified ring is reset to its default size. 3b44b

Argument/result types: 3b44c

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

ringtype - INTEGER[RETURN=62/FILERETURN=67] 3b44c1

count - LSEL*(INTEGER)/EMPTY 3b44c2

Set/reset the startup branch option 3b45

STARTUP (branch) 3b45a

This procedure sets the address of a command branch BRANCH to be executed at process startup time. If BRANCH is EMPTY, the startup branch option is reset to its default value so that no startup branch is executed. 3b45b

Argument/result types: 3b45c

branch - LSEL*(LINK)/EMPTY 3b45c1

Set/reset the viewspecs option 3b46

VIEWSPECS (viewspecs) 3b46a

This procedure sets the standard viewspecs to have the value VIEWSPECS. If VIEWSPECS is EMPTY, the viewspecs are reset to their default value. 3b46b

Argument/result types: 3b46c

viewspecs - VIEWSPEC*/EMPTY 3b46c1

(ISI) Do you need Show Directory and/or Show Archive?

Show the specified property 3b47

SHOW (property, type, diroptions => message) 3b47a

This procedure returns in MESSAGE the value of PROPERTY (possibly further specified by TYPE) including (for PROPERTY equal DIRECTORY) only the DIROPTIONS specified. The properties that can be shown are file status, file default directory for links, file modifications status, file return ring, file size, statement return ring, archive directory, various directory diroptions, disk space status, name delimiters, viewspecs, the status of the user programs buffer, and the status of any or all of the tool's options. 3b47b

Argument/result types: 3b47c

property -

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

```
KEY2*(FILE=51/RETURN=62/ARCHIVE=61/DIRECTORY=9/
DISK=52/NAME=18/VIEWSPECS=57/BUFFER=58/OPTION=59]                    3b47c1

   FILE:                          type - INTEGER[STATUS=52/

   DEFAULT=54/MODIFICATIONS=53/

   RETURN=62/SIZE=55/MARKER=51]
                                  diroptions - EMPTY                 3b47c1a

   ARCHIVE:                       type -
   BOOLEAN/LSEL*(NAME)
                                  diroptions - EMPTY                 3b47c1b

   RETURN/DISK/VIEWSPECS/BUFFER:
                                  type - EMPTY
                                  diroptions - EMPTY                 3b47c1c

   DIRECTORY:                     type -
   BOOLEAN/LSEL*(DIRECTORY)
                                  diroptions -
   DIROPTIONS*                                                       3b47c1d

   NAME:                      type - DSEL*(STATEMENT)
                                  diroptions - EMPTY                 3b47c1e

   OPTION:                        type - INTEGER[ALL=52/

   CURCONTEXT=51/DEFAULT=54/

   EXTERNAL=59/LEVELADJUST=53/

   NAME=18/PRINTOPTIONS=55/
                                  RETURN=62/STARTUP=56/
                                  VIEWSPECS=57]
                                  diroptions - EMPTY                 3b47c1f

   message - CHARSTR                                                 3b47c2

Simulate terminal type                                              3b48

   SIMULATE (terminalclass)                                         3b48a

   This procedure records the terminal class being simulated,
   as well as certain properties of that class.  It is also
   called when the user links to another terminal.                  3b48b

   Argument/result types:                                           3b48c
```

terminalclass - INTEGER[TYPEWRITER=52/HALFDUPLEX=53/
LINEATATIME=54/CHARATATIME=55/DISPLAY=51] 3b48c1

Sort statements 3b49

SORT (location) 3b49a

This procedure sorts the statements at LOCATION according to the currently established sort key program. 3b49b

The curmkr is set to the head of the sorted set of statements. 3b49c

Argument/result types: 3b49d

location - DSEL*(GROUP/BRANCH/PLEX) 3b49d1

Substitute a value throughout a structure 3b50

SUBSTITUTE (destination, subpairs, filter) 3b50a

This procedure substitutes the values SUBPAIRS (a list of pairs of the form oldtextentity,newtextentity) in the structure located at DESTINATION. Only those statements selected by FILTER are treated. 3b50b

The curmkr is not changed. 3b50c

Argument/result types: 3b50d

destination - DSEL*(STRUCENT*) 3b50d1

subpairs - LIST(LIST(LSEL*(TEXTENTITY),
LSEL*(TEXTENTITY))) 3b50d2

filter - VIEWSPEC*/EMPTY 3b50d3

Transpose two entities 3b51

TRANSPOSE (location1, location2, filter) 3b51a

This procedure transposes the two entities at LOCATION1 and LOCATION2. Only those statements selected by FILTER are affected. 3b51b

The curmkr is set to LOCATION1. 3b51c

Argument/result types: 3b51d

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

location1 - DSEL*(TEXTENTITY*/STRUCENT*) 3b51d1

location2 - DSEL*(TEXTENTITY*/STRUCENT*) 3b51d2

filter - VIEWSPEC* 3b51d3

(ISI) Do you need this facility?
Trim a directory 3b52

TRIM (count => filelist) 3b52a

This procedure trims the connected (TENEX) directory , deleting all but COUNT versions of each file. It returns a list of character strings, FILELIST, containing the names of trimmed files. 3b52b

Argument/result types: 3b52c

count - LSEL*(INTEGER) 3b52c1

filelist - LIST(CHARSTR) 3b52c2

(ISI) Do you need Undelete File?
Undelete a file or modifications 3b53

UNDELETE (type, filename) 3b53a

This procedure undoes a previous deletion of TYPE (a file or modifications to a file) specified by FILENAME. 3b53b

Argument/result types: 3b53c

type - INTEGER[FILE=51/MODIFICATIONS=53] 3b53c1

FILE: filename - LSEL*(OLDFILENAME) 3b53c1a

MODIFICATIONS: filename - EMPTY 3b53c1b

(ISI) Do you need this facility?
Update a file 3b54

UPDATE (updatetype, filename, newname) 3b54a

This procedure performs an UPDATETYPE update on the file specified by FILENAME. If it is being renamed, NEWNAME specifies the new name. 3b54b

Argument/result types: 3b54c

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

updatetype - INTEGER[NEW=51/OLD=52/COMPACT=53/RENAME=54] 3b54c1

NEW/OLD/COMPACT: newname - EMPTY 3b54c1a

RENAME: newname - LSEL*(NEWFILENAME) 3b54c1b

filename - LSEL*(OLDFILENAME) 3b54c2

(ISI) Do you need this facility?
Verify a file 3b55

VERIFY () 3b55a

This procedure checks the internal structure of the loaded file and performs a normal or aborted return depending on whether the file is good or bad. 3b55b

EKM 5-FEB-75 20:27 25307

Definition of PCP Callable Routine in the NLS Editor - Primarily for ISI

(J25307) 5-FEB-75 20:27;;;; Title: Author(s): Elizabeth K. Michael/EKM; Distribution: /NPG([ACTION]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: EKM; Origin: < NSW-SOURCES, ISI-DOC.NLS;2, >, 5-FEB-75 16:42 KJM ;;;;####;

DIA 6-FEB-75 08:04 25308

Extensions to the L10 Programming Language for the DEC PDP-10 and DEC PDP-11

This document should accompany the L10 Documentation entitled A Programming Laguage for the Augmentation Research Center by W.H. Paxton. Offline copies of both are available in Rm. J2082.

Extensions to the L10 Programming Language for the DEC PDP-10 and DEC PDP-11

Title 1

Extensions to the L10 Programming Language for the DEC PDP-10 and DEC PDP-11 1a

Abstract 2

L10, a system programming language for the DEC PDP-10, was developed and implemented at SRI-ARC in 1971. Recently, the language has been expanded and improved, and a compiler for the DEC PDP-11 has been implemented. With few exceptions, L10 procedures written for the PDP-10 can be compiled and run on the PDP-11 with the same results, and vice versa. This document describes the language changes and should accompany the original L10 manual. 2a

Introduction 3

We have written a cross compiler for the L10 language for the PDP-11. The compiler is called L1011 and runs on a TENEX. 3a

We have made several changes in the L10 language. Primarily, we have added to the declaration syntax, added coroutines, and improved signals. In most cases, programs using the old L10 definition will not have to be changed. 3b

The new version of the L10 compiler (for PDP-10) is available for use as <SUBSYS>XL10. 3c

This document serves as a "differences" manual, in which changes are broken down into additions, deletions, and syntax changes. The L10 definition document will be updated as soon as time permits, but this document should remain useful to those familiar with the old L10. 3d

Additions: 4

In DECLARES's 4a

Values can be expressions 4a1

Initial values to be stored in declared items can be of the same syntax as expressions, with the exception of the CASE expression and the bit-AND/OR/XOR functions (i.e. .A .V and .X). All items in the expression must however be defined by the time that declaration is compiled (one pass compilation, you know). 4a1a

Also, the symbol "=" to assign a value to a declared variable may be replaced with the symbol "_". They are equivalent. The symbol "_" is not allowed for declarations of symbols that are not variable, as in CONSTANT, ADDRESS and REGISTER declarations. 4a1b

The intent is to use "_" for variable symbols and "=" for constant symbols. 4a1b1

Use ADDRESS, not SET 4a2

As a more descriptive declaration, use the word ADDRESS where you now use SET. "SET" is still accepted, however. 4a2a

The "SET" syntax may be phased out in the future. 4a2b

New declaration: DECLARE CONSTANT 4a3

Symbols can be declared to represent a constant value (given by an expression). The symbols are then used syntactically as though they were variables containing the specified value, but they take up no memory and the compiler takes advantage of the situation where possible. 4a3a

External constants function just like non-external constants for the file they are declared in, but they do take up memory and are available to other code-files. Incidently, there is no runtime code produced to insure that code in other files does not store into a n external constant, unless it is in a write protected page. 4a3b

Declared items may be NLS names (like procedures and labels) 4a4

An alternate form for declaring symbols has been included. The general form is: 4a4a

'(,ID ') declareword ';' 4a4a1

The declareword may be preceeded by "EXTERNAL" if desired. Definitions follow: 4a4b

declareword = 4a4b1

"STRING" 4a4b1a

which must be followed by '[expression '] or 4a4b1a1

('= / '_) ,SR for actual initial value. 4a4b1a2

"CONSTANT" '= expression 4a4b1b

"TEXT" "POINTER" 4a4b1c

"STACK" '[expression ['; expression] '] 4a4b1d

"RING" '[expression ['; expression] '] 4a4b1e

"ADDRESS" '= expression 4a4b1f

"DECLARE" 4a4b1g

the identifier may be followed by '[expression '] for array dimension or 4a4b1g1

('= / '_) expression for actual initial value or 4a4b1g2

('= / '_) '(#<',> itemval ') for a list of values for an array. 4a4b1g3

or the declareword may be absent, 4a4b1h

which is the same as writing "DECLARE". 4a4b1h1

itemval = 4a4b2

,SR to get that string into memory (not an A-string) 4a4b2a

(In this case the string is packed into words starting with the current word. The length is not stored. An a-string consists of two ADDRESSes followed by the packed string. The first address contains the max number of characters in the string (length) and the second contains the current length. The two addresses occupy one word on the PDP-10 and two words on the PDP-11.) 4a4b2a1

'S ,SR to get the address of that a-string 4a4b2b

'S ,ID to get the address of that symbol 4a4b2c

or an expression (defined of course). 4a4b2d

Example: 4a4b3

(X) CONSTANT = 5; 4a4b3a

(list) _ (25,3*X,$list,S"string"); 4a4b3b

New type (REF, etc.) designation 4a5

Where a REF or POINTER symbol is declared, it can be designated a REF or POINTER type symbol in the declaration statement. 4a5a

For example: 4a5a1

```
DECLARE x REF _ $y;    or                    4a5a1a

(x) REF _ $y;                                4a5a1b
```

are equivalent to: 4a5a2

```
DECLARE x = $y;                              4a5a2a

REF x;                                       4a5a2b
```

Both define x to be a REF variable with initial value set to the address of y. 4a5a3

Similarly, a variable that is used as record pointer can be declared to be a pointer to the record thus: 4a5b

```
(x) RECPTR recnam;                           4a5b1
```

Where recnam is the name of a record. This example declares x to be a single-word variable AND signals the compiler (and reader) that it will always be used as a pointer to a record of the form defined in recnam. The RECPTR designation allows the use of the x,SIZE construct (see below). 4a5b2

The modifier EXTERNAL should preceed the type designation: 4a5c

```
(x) EXTERNAL REF _ $y;                       4a5c1
```

In Procedure LOCAL's 4b

LOCAL CONSTANT is allowed 4b1

The constant symbol is a local but takes up no memory. The symbol is available for other LOCAL use at the end of the procedure, as are other locals. 4b1a

Runtime assignment is allowed at LOCAL declaration time 4b2

An initial value for locals may be specified. The syntax is

the same as assignment of values to declared symbols, but the expression may be of the most general form and is evaluated at runtime (compile time expressions are resolved to a single constant, of course). 4b2a

It is equivalent to declare a local to be an expression, and to store an expression in that local at the start of the procedure. 4b2b

LOCAL declared items may be NLS names 4b3

The same '(,ID ') declaration syntax is allowed inside procedures with the following exceptions: 4b3a

The word LOCAL is used instead of the word DECLARE. 4b3a1

Those symbols may not be EXTERNAL. 4b3a2

STACK and RING are not implemented (this may be added later). 4b3a3

REF and POINTER designation in LOCAL declarations 4b4

The designators REF, POINTER or RECPTR <name> may follow local symbol declarations just as in the DECLARE statement. 4b4a

Note that a symbol may be both a REF and a RECPTR. The correct order is: 4b4a1

(x) REF RECPTR recnam; 4b4a1a

In FORMAL arguments 4c

you may specify the type of argument 4c1

Symbols of type REF, POINTER or RECPTR <name> can be specified to be so, right in the formal argument list. For example: 4c1a

(p) PROCEDURE (arg1 REF, arg2 POINTER); 4c1a1

This is equivalent to saying the following after the procedure heading: 4c1b

REF arg1; 4c1b1

POINTER arg2; 4c1b2

In buildin field names 4d

A new builtin uppercase field name exists: SIZE. The expression 4d1

```
x.SIZE                                                    4d1a
```

has the value equal to the number of WORDS in the specified record. The symbol x must be either a record name, or a record pointer which appears with a RECPTR modifier somewhere before the x.SIZE expression is used. 4d2

For example, procedure P is passed pointer a to record RECNAM and wants to store the SIZE of RECNAM in its local variable N: 4d3

```
(p) PROCEDURE (x RECPTR recnam);                          4d3a
    LOCAL n;                                              4d3a1
    ...                                                   4d3a2
    n _ x.SIZE;                                           4d3a3
    ...                                                   4d3a4
```

The expression x.SIZE is a compile-time expression provided x is a record name and the record is defined before the x.SIZE expression appears. The following example declares a record and an array that will hold several records: 4d4

```
DECLARE CONSTANT n=5; % currently 5 records in rec %      4d4a
(recnam) RECORD                                           4d4b
    fieldi[10],                                           4d4b1
    ...                                                   4d4b2
    fieldn[4];                                            4d4b3
DECLARE rec[recnam,SIZE*n];                               4d4c
    % an array large enough to hold n recnam records %    4d4c1
```

In RECORD definitions 4e

Field definitions within a record definition have been extended to allow 4e1

(1) a compile-time expression to specify the number of bits in the field, or 4e1a

(2) a number of sub-fields to specify the number of bits in the field. 4e1b

For example, the following defines a one word PDP-10 record with the field "word" being the whole word, "adr" as the right half, "indx" as the index field, "half" as the left half, "indir" as the indirect bit, "a" as the accumulator field and "op" as the opcode field. 4e2

(wordrec) RECORD 4e2a

word[adr[18], half[indx[4], indir[1], a[4], op[9]]; 4e2a1

(Remember that bits are assigned from LSB to MSB). 4e2a2

The expression x.indir will obtain the indirect bit for any word addressed by x. Likewise x.half will obtain the left half. 4e2b

In primary symbols 4f

several new symbols builtin 4f1

The new uppercase identifiers are to help out when programming both the PDP-10 and PDP-11. They are essentially constants that are builtin (all numbers decimal): 4f1a

WORD = 36 on PDP-10, 16 on PDP-11 4f1a1

ADDRESS or ADDR = 18 on PDP-10, 16 on PDP-11 4f1a2

CHARACTER or CHAR = 7 on PDP-10, 8 on PDP-11 4f1a3

CPU = 10 on PDP-10, 11 on PDP-11 4f1a4

Register names are now built in: 4f2

R0 through R15 on PDP-10, R0 through R7 on PDP-11, designate registers (decimal numbers). 4f2a

S designates the stack register; 4f2b

M designates the mark register. 4f2c

PC and P designates register 7 on the PDP-11 only. 4f2d

A1 through A4 on PDP-10 only, designate L10 scratch accumulators. 4f2e

On PDP-10, S=R15, M=R14, A1-A4=R10,R13. 4f2f

Program labels are now LOCAL to a procedure: 4f3

Labels are LOCAL to a procedure unless they appear in an EXTERNAL statement before they are defined. 4f3a

Source File name 4g

(Pending modification to NLS) 4g1

Every File compiled will have a symbol "sfilev" (source-file-version) included in the symbol table. It will be the adddress of an a-string which is the entire source file name including version number. 4g2

In addition to the obvious usefulness of this, we may use it later in source language debugging. 4g3

Success/Fail RETURN 4h

We have added syntax to L10 to return a success or fail condition in addition to regular procedure call results. The previous return syntax now implies a successful return. 4h1

This is nothing more than providing one extra return result which is always there, and is set to non-zero (TRUE) if not otherwise mentioned. 4h2

The boolean value (success = TRUE, failure = FALSE) is given inside square brackets after the word RETURN. For example: 4h3

```
RETURN [FALSE]; % a fail return with no args %
```
4h3a

```
RETURN [X] (Y); % success/fail depends on x, one argument returned %
```
4h3b

```
RETURN; % success return with no args %
```
4h3c

To the calling procedure, the boolean value looks like a multiple result, but is indicated by writing it inside square brackets. It can be stored as in these examples: 4h4

```
a _ proc1(arg1,arg2; [p],b);                                        4h4a

   % if proc1's return looks like RETURN [x] (res1,res2);           4h4a1

   first result (res1) goes into a, second (res2) into b, x
   goes into p %                                                    4h4a2

proc2( :[f] );                                                      4h4b

   % proc2 to has no args or results, but we find out if it
   fails or succeeds by looking at f %                              4h4b1
```

A calling procedure need not store the success/fail value if it does not need it. Likewise, it can always store the value -- it will be TRUE if the RETURN statement does not specify a value. 4h5

Syntax: 4h6

```
return =                                                            4h6a

   "RETURN" [ '[ expression '] ] [ '( expressionlist ')             4h6a1

procedurecall =                                                     4h6b

   fwlhs [ '( args ') ]                                             4h6b1

args =                                                              4h6c

   arglist ': [ '[ fwlhs '] ] resultlist                            4h6c1

fwlhs = full-word-left-hand-side                                    4h6d
```

NOTE: in the above syntax, the catchphrase invocation option is deliberately omitted for clarity. See SIGNALS for complete syntax. 4h6e

Coroutines 4i

We have implemented a rather general type of coroutine linkage. 4i1

In the following discussion, the word 'routine' will be used to mean procedure or coroutine. 4i2

Introduction 4i3

As described below, a procedure establishes coroutine links

and then starts communication over 'ports' via port-calls (PCALL). 4i3a

A 'port' is a handle for an instance of a coroutine and is used to indicate the routine that is to receive the PCALL. 4i3b

Information is passed by value as are procedure arguments. As two routines PCALL back and forth, the arguments passed by one look like results to the other. 4i3c

Coroutine definition and scope 4i4

Coroutines are defined just like procedures, except the word COROUTINE is used instead of PROCEDURE. 4i4a

Coroutines have the following limitations: 4i4a1

A Coroutine cannot contain a RETURN statement. 4i4a1a

A Coroutine's first statement must be a PORT ENTRY statement. 4i4a1b

A coroutine cannot contain an OPENPORT statement, except in the PORT ENTRY statement. 4i4a1c

OPENPORT and PORTENTRY will be described later. 4i4a2

In L10, each runtime instance of a procedure 'owns' some stack area (called a frame) for its LOCALS and for intermediate results it may have to push on the stack. 4i4b

When a procedure returns, the stack frame is gone forever. 4i4b1

Coroutines have associated frames also, but DO NOT own their stack frame. It is owned by the procedure that called them with an OPENPORT statement. If the caller was a coroutine, then the stack frame is owned by the procedure that owns that coroutine. 4i4c

As a result, when a procedure returns, all coroutine instances which it owned vanish with it. 4i4d

Ports 4i5

A procedure (for example, A) establishes a coroutine link by doing an OPENPORT statement such as 4i5a

```
OPENPORT b(arg1,arg2:[p]);                                    4i5a1
```

B must be a coroutine. Two formal arguments are passed in this example. The result stored in p is not a sucess/fail value but a port identifier. The procedure will use that port identifier in subsequent port-calls. The success/fail option does not exist for coroutines or port-calls. 4i5a2

The stack frame for B belongs to A, and will not disappear until the procedure A returns. Hence the port ID in p is good throughout the procedure's life, and may even be passed to other routines that it calls. 4i5a3

Furthermore, any ports opened as a result of the OPENPORT on B also belong to the procedure A. 4i5a4

Writing Coroutines 4i6

The first statement in a coroutine is a PORT ENTRY statement. It is used to initialize a coroutine instance and establish the coroutine linkage. In general, the coroutine does initialization in the PORT ENTRY statement, and PCALL's back to the routine doing the OPENPORT without performing its intended function. 4i6a

The formal arguments and LOCALS in a coroutine instance remain unchanged and local to that instance, as though control had not left that coroutine. When the owning procedure returns, that information is lost. 4i6b

The simplest form of the PORT ENTRY statement is: 4i6c

```
PORT ENTRY EXIT PCALL;                                        4i6c1
```

Control goes to the following statement when the first PCALL to the coroutine is made. 4i6d

A more general entry statement is: 4i6e

```
PORT ENTRY [p]                                                4i6e1
  BEGIN                                                       4i6e1a
  ... % initialization %                                      4i6e1b
  END                                                         4i6e1c
```

EXIT r _ PCALL (arg1,arg2,arg3,arg4:r2,r3,r4) 4i6e2

The EXIT phrase is the PCALL back to the routine doing the OPENPORT. This example returns four arguments (arg1-arg4) to the OPENPORT statement (max allowed = 4). 4i6e3

Results may also be specified in the EXIT phrase (here, r1 through r4). Such results are stored from the arguments given in the first PCALL to this coroutine. 4i6e4

The [p] phrase is optional. It is used to save the port id of the routine that did the OPENPORT. The port id of the routine that did the openport is ALWAYS stored in PORT when the PORT ENTRY statement is executed (see explaination of PORT below). 4i6e5

Note that a coroutine has both formal arguments (available for the duration of the coroutine instance) and PCALL arguments -- which take the form of results of PCALLs and must be stored in local or global variables. 4i6e6

The PCALL port-call syntax is 4i6f

PCALL [p] (args : [rp] res) 4i6f1

P is the port id (to be called). 4i6f2

The [p] phrase is optional. If not present, the value in PORT will be used (see below). 4i6f2a

RP is the port id for the returning routine. 4i6f3

The [rp] phrase is optional. If not present, the returning port id will be stored in PORT. 4i6f3a

Up to 4 arguments and Results are allowed. The PCALL is an expression whose value is the first result. 4i6f4

Inside a coroutine, if the returning port id is not explicitly saved, it is implicitly saved in a coroutine-local predefined variable called PORT. 4i6g

Inside a procedure, if the returning port id is not explicitly saved, it is lost forever. References to 'PORT' are not allowed in procedures. 4i6h

Inside a coroutine, if the port to be called is not specified, the port id in PORT is used. 4i6i

Inside a procedure, the port id of the port to be called must always be specified. 4i6j

Be advised that if the port is explicitly saved in a coroutine, the value in PORT is NOT CHANGED. 4i6k

Note that a PCALL always comes back (signals excepted), but not necessarily from the same port. Hence it may be desirable to save the returning port number under some conditions. 4i6l

Here follow some phoney examples: 4i7

PROC opens C1 which provides a sequence of integers. PROC then port-calls C1 in a loop, handing the integers to UGH. The port P serves as an infinite source of integers. 4i7a

```
(proc) PROCEDURE;                                          4i7a1

   (p) ; % port id %                                       4i7a1a

   OPENPORT c1( 3 ;[p] ); % get port id %                  4i7a1b

   LOOP ugh( PCALL [p] ) ;                                 4i7a1c

   END.                                                    4i7a1d

(c1) COROUTINE % initial value for sequence %              4i7a2

   (init); % initial integer passed as formal in openport
   %                                                       4i7a2a

   PORT ENTRY EXIT PCALL;                                  4i7a2b

   LOOP PCALL (init:=init+1);                              4i7a2c

   END.                                                    4i7a2d
```

Here, POOH opens C2, which opens C3. C2 passes the port id for POOH to C3. POOH unknowlingly port-calls C2 to get integers. C2 always port-calls C3 with a number (x) and C3 port-calls back to POOH with x*2. POOH doesn't care who returned the call -- he calls C2 again. 4i7b

```
(pooh) PROCEDURE;                                          4i7b1

   % locals %                                              4i7b1a
```

```
        (p); % port id %                                          4i7b1a1
      OPENPORT c2( :[p] );                                        4i7b1b
      LOOP ugh( PCALL [p] );                                      4i7b1c
    (c2) COROUTINE;                                               4i7b2
      % locals %                                                  4i7b2a
        (x) _ 21;                                                 4i7b2a1
        (p) ; % port id for c3 %                                  4i7b2a2
      PORT ENTRY                                                  4i7b2b
        OPENPORT c3(PORT :[p] )                                   4i7b2b1
          % PORT is formal arg for C3 - p is port id of C3
          %                                                       4i7b2b1a
      EXIT PCALL;                                                 4i7b2c
      LOOP PCALL [p](x_x+1);                                      4i7b2d
      END;                                                        4i7b2e
    (c3) COROUTINE                                                4i7b3
      % formals %                                                 4i7b3a
        (q); % port id for pooh %                                 4i7b3a1
      % locals %                                                  4i7b3b
        (z); % PCALL result from C2 %                             4i7b3b1
      PORT ENTRY EXIT z _ PCALL ();                               4i7b3c
        % z is arg that C2 passes in first PCALL [p](x_x+1)
        %                                                         4i7b3c1
      LOOP PCALL [q] (z*2);                                       4i7b3d
      END;                                                        4i7b3e
Syntax:                                                           4i8
```

```
coroutine =                                                    4i8a
   '( .ID ') "COROUTINE"                                       4i8a1
   formals ';                                                  4i8a2
   locals                                                      4i8a3
   "PORT" "ENTRY" portdes [';]                                 4i8a4
      [ statement ] [';]                                       4i8a4a
      "EXIT" [ lhs '_ ] "PCALL" pcall2 ';                      4i8a4b
   procedurebody                                               4i8a5
   "END."                                                      4i8a6
openport =                                                     4i8b
   "OPEN" "PORT" [ lhs '_ ] procedurecall                      4i8b1
pcall =                                                        4i8c
   "PCALL" pcall2                                              4i8c1
pcall2 =                                                       4i8d
   portdes  [ '( args ') ]                                     4i8d1
      % here, portdes is optional in coroutines, required in
      procedures %                                             4i8d1a
portdes =                                                      4i8e
   [ '[ ( fwlhs / "PORT" ) '] ]                                4i8e1
args =                                                         4i8f
   arglist ': [ '[ fwlhs '] ] resultlist                       4i8f1
```

Deletions: 5

Special syntax of the following forms have been deleted because they are no longer used or desirable: 5a

DSP ... 5a1

Only 4 return results are allowed now. Also, only 4 PCALL arguments/results are allowed. It is assumed that greater amounts of information will be passed via a pointer to a record or list of variables. 6a1

System interface (JSYS, register, opcode) constructs 6b

System calls, assembly instructions and register references will not be allowed unless the code-file has the following statement before any such reference: 6b1

ALLOW! 6b1a

The ALLOW! statement belongs at the procedure-heading level and should be the first thing in the file. 6b2

The purpose of this restriction is to help us isolate system-interface code and maintain system-interface independent code in certain files. 6b3

String subscript references 6c

If a string element is referenced as follows 6c1

string[i] 6c1a

the Ith character of the string is returned. Previously, if the index I were not within the range [0,M] where M is the max length of the string, then an ENDCHR was returned. This feature was almost never used and was somewhat expensive, and hence has been removed. String subscript references are now

faster, but wild indices do the same thing that wild array indices do - namely, get you garbage. 6c2

Storing into a string element with a wild index is as serious an error as storing into an array with a wild index - program or data areas may be destroyed or memory violations may occur. 6c3

If you wish to have end conditions recognized for you, use the READC constructs to read from a string. READC is quite fast once the position within the string has been established. It is not, however, faster than string subscripts. 6c4

String constructions of the form *string* _ <list of characters, strings or substrings> will check for end conditions. 6c5

Signals 6d

The operation of signals has been changed significantly. The old syntax no longer applies. 6d1

Introduction 6d2

A signal is a way for a routine to communicate (via transfer of control and arguments) to other routines in the thread of control. Their principle use is in handling error conditions or unusual circumstances. This allows the expression of error-free behavior separately from error handling and thus permits clearer, more readable code. 6d2a

The thread of control includes all procedure instances that have not returned, in the order that those instances first had control. Also included in the thread of control are coroutines for which the owning procedure has not returnd. 6d2b

The exact order in the thread of control is the cronological order in which (1) procedures were called, or (2) coroutines were OPENPORTed. Of course, procedures that have returned, or coroutines that belong to procedures that have returned, are no longer in the thread of control. 6d2b1

Various types of signals can be generated. All of them pass arguments. The first argument is by convention always a signal name. A signal is characterized by signal name and signal type. 6d2c

A routine gets control when a signal is generated, if it

invoked a catchphrase and if that catchphrase is eligible. We will then say that the catchphrase has been activated. 6d2d

A catchphrase is a block of code that conforms to special syntax. 6d2e

The catchphrase contains special code to identify and handle signal situations. We will call this dispatching the signal. 6d2f

Signal names 6d3

Several signal names are predefined and have specific meanings. Other names can be originated by the programmer. 6d3a

Predefined names are CONSTANTs with value greater than 10000 octal. Programmer-defined names should be between 1 and 7777 octal. 6d3b

The predefined signal names and meanings are listed at the end of this document. 6d3c

Types of signals 6d4

There are three types of signals: 6d4a

ABORT(signame,a2,a3,a4) 6d4a1

An abort is used where a routine cannot continue for some reason. 6d4a1a

Up to four arguments are allowed. The first argument must be the signal name. The second argument is by convention the address of a user-readable string that describes the problem. 6d4a1b

res1 _ HELP(signame,a2,a3,a4:res2,res3,res4) 6d4a2

A help is used to obtain help from a routine up in the thread of control somewhere. Generally, but not necessarily, control is returned and results may be returned. 6d4a2a

Up to four arguments are allowed. Up to four results may be returned by the catchphrase that dispatched the signal. 6d4a2b

NOTE(signame,a2,a3,a4) 6d4a3

A note is used to pass information up to any number of routines in the thread of control. All eligible catchphrases will be activated with this signal. Control will always be returned, no results will be returned. 6d4a3a

Dispatching signals 6d5

Catchphases may do arbirary computing except PCALL's, GOTO's or RETURN's, but they must finish by doing something with the signal. There are three ways it may dispatch the signal. Each is an unconditional transfer of control. 6d5a

CONTINUE; 6d5a1

A continue means to pass the signal on up the thread of control to other catchphrases. Arbitrary computing may be done before the CONTINUE. Any of the three types of signals can be CONTINUEd. CONTINUE is the only way to handle a NOTE. 6d5a1a

After executing a CONTINUE, another catchphrase will be activated and the signal will still be in progress. 6d5a1b

RESUME(res1,res2,res3,res4); 6d5a2

A resume means to return control to the signalling routine. Up to four arguments may be returned. A resume is only allowed for a signal of type HELP. By convention, the first result indicates whether or not help has been given (gothelp or nohelp). 6d5a2a

After executing a RESUME, control will be returned to the signalling routine and the signal will no longer be in progress. 6d5a2b

TERMINATE; 6d5a3

A TERMINATE terminates the signal and gives control to the routine that caught the signal. In particular, control goes to the routine that invoked the catchphrase that did the TERMINATE. Control is transferred to the location specified when the catchphrase was invoked. 6d5a3a

TERMINATE is VERY SERIOUS because it alters the structure of the thread of control and other runtime state information: 6d5a3b

All instances of routines (procedures and coroutines) in the thread of control below the catching routine are notified (via NOTE(unwind)) and then those instances are ERASED. 6d5a3b1

All instances of catchphrases invoked by the erased routines are also lost. 6d5a3b2

In the case that signals are nested (i.e. a signal within a signal), the outter signals are deactivated. That is, after the TERMINATE, no signals are in progress and state information pertaining to all the signals is lost. 6d5a3b3

With respect to coroutines, the state of the routine (say A) whose catchphrase does the TERMINATE is restored to its state at the time of the INVOKE. 6d5a3c

This means that any ports opened AFTER the INVOKE takes place will be lost when routine A's catchphrase does a TERMINATE. 6d5a3c1

For this reason, it is generally a good idea to open all ports before invoking catchphrases. 6d5a3c2

A TERMINATE does not alter the state of catchphrases for the routine whose catchphrase does the TERMINATE. 6d5a3d

Note that a catchphrase may generate a signal as part of its signal processing. This results in a nested signal situation which is allowed within runtime-package memory constraints. (Currently set to 5). 6d5b

IF the catchphrase does a NOTE or HELP, when control returns to the catchphrase the values of SIGNAL and SIGNALTYPE are restored. 6d5b1

However, that same catchphrase would be activated for the signal (its own signal). If it stored signal arguments in local variables, the previous values of those variables would be lost. 6d5b2

But of course, a catchphrase can disable itself, generate a signal, enable itself again and then dispatch the signal at hand. 6d5b3

Catchphrases 6d6

Catchphrases are defined as follows: 6d6a

'(name ') "CATCHPHRASE" "(:" results ') '; 6d6a1

statement 6d6a1a

results = <places to store results 2 3 and 4> 6d6a2

The first result is stored in the runtime-package global referred to as SIGNAL. 6d6a2a

Catchphrase names (and thus catchphrases) are global (if defined at the procedure level) or local to a routine (if defined within a routine). 6d6a3

Local catchphrases may only be invoked by the routine to which they are local. 6d6a4

A catchphrase may appear anywhere within a routine. By convention they should appear at the end (before the END.). 6d6a5

Catchphrases are not executed inline. They serve as declarations but unlike local declarations, they may appear anywhere. 6d6a6

The single statement in the catchphrase is usually a CASE statement based on the signal name which is referred to as SIGNAL. The catchphrase code can also test SIGNALTYPE to find out which of the three types of signals is at hand. 6d6b

SIGNALTYPE=helptype in the case of a HELP 6d6b1

SIGNALTYPE=aborttype in the case of a ABORT 6d6b2

SIGNALTYPE=notetype in the case of a NOTE 6d6b3

A catchphrase is invoked in the following way: 6d6c

INVOKE(name,label); 6d6c1

Where "name" is the name of the catchphrase and "label" is a label in the invoking routine to which control will go if the catchphrase TERMINATEs the signal. 6d6c2

The label need not be specified if the catchphrase will never do a TERMINATE. A runtime error will occur if no label is specified and a TERMINATE is done. 6d6c3

As a convenience, the label may be replaced by syntax to indicate that the routine should return (success or fail) upon the TERMINATE: 6d6c4

INVOKE(catch,RETURN) or 6d6c4a

INVOKE(catch,RETURN[TRUE]) or 6d6c4b

INVOKE(catch,RETURN[FALSE]) 6d6c4c

Note that in these cases, NO arguments are returned. Of course this feature may not be used in coroutines, as they may not return. 6d6c4d

A routine may invoke any number of catchphrases. They will be activated, if eligible, in the inverse order of invocation. 6d6c5

When a procedure returns, all catchphrases invoked by it and its coroutines are effectively dropped (de-invoked). 6d6c6

A catchphrase is dropped (de-invoked) in the following way: 6d6d

DROP(name); 6d6d1

This effectively un-does the INVOKE(name). The situation is as if the catchphrase had not been invoked. 6d6d2

If the routine instance doing the DROP has not invoked a catchphrase named NAME, then the operation is a NO-OP. 6d6d3

If the routine instance has invoked several catchphrases named NAME, then the most recently invoked one is DROPped. 6d6d4

A catchphrase may be enabled and disabled as follows: 6d6e

ENABLE(name); 6d6e1

DISABLE(name); 6d6e2

Note that enable/disable increment and decrement a counter. The INVOKE sets the count to one. 6d6e3

If the routine instance doing the ENABLE or DISABLE has not invoked a catchphrase named NAME, then the operation is a NO-OP. 6d6e4

If the routine instance has invoked several catchphrases named NAME, then the most recently invoked one is ENABLEd/DISABLEd. 6d6e5

A catchphrase is eligible to be activated on any signal if: 6d6f

It has been invoked and its enable count is >0 AND 6d6f1

the invoking routine's position in the thread of control is ABOVE or EQUAL to the position of the routine generating the signal. 6d6f2

Note that this means that a routine can catch its own signals. 6d6f3

As a convienence, a catchphrase may be invoked over one procedure call by naming the catchphrase inside the procedure call syntax as follows: 6d6g

procall= fwlhs '(args [': results] ['; catchname] ') 6d6g1

If that catchphrase TERMINATEs the signal, control will be passed back to the procedure call statement just as if the procedure had returned. In that case, results from the procedure will be undefined. 6d6g2

Runtime package signals 6d7

Several rumtine-package generated signals are of interest to the programmer: 6d7a

return 6d7a1

In the event that a procedure or one of its coroutines has an invoked catchphrase when the procedure returns, a signal named "return" is generated. All catchphrases invoked by the procedure or any of its coroutines is eligible (if enabled). 6d7a1a

This allows cleanup code that must be executed when the routine goes away to reside in one place - the catchphrase. 6d7a1b

It is also the only automatic way that a coroutine is notified that it will go away. 6d7a1c

There is some expense associated with this operation.

It can be avoided by DROPping all catchphrases before doing the RETURN. 6d7a1d

unwind 6d7a2

When a catchphrase does a TERMINATE, all routines below it in the thread of control will vanish. They are notified of this first by way of an "unwind" signal of type NOTE. 6d7a2a

They cannot do anything to prevent their vanishing, but they can do arbitrary computing to prepair for it. 6d7a2b

In the event of a serious runtime system error, all eligible catchphrases will get the "unwind" NOTE. Such an event can be detected because the runtime-package global sysrip (recover in progress) will be TRUE. 6d7a2c

saroverflow 6d7a3

String construction constructs in the language result in calls to runtime routines that build the strings in a global area called SAR. If that area overflows an ABORT(saroverflow,$"string too long") occurs. 6d7a3a

stkoverflow and stkunderflow 6d7a4

When PUSH and POP operations are done on programmer defined stacks, checks are made for under/overflow. An ABORT(stkunderflow,$"stack underflow") or ABORT(stkoverflow,$"stack overflow") may occur. 6d7a4a

nohelp and gothelp 6d7a5

If a routine does a HELP(...) and no catchphrases dispatch the signal with a RESUME, the signal is resumed with a first result of "nohelp". By convention, all catchphrases that RESUME should pass "gothelp" as the first argument. The names "nohelp" and "gothelp" are predefined symbols as are signal names in this list. 6d7a5a

stringoverflow and changestring 6d7a6

In the event that a programmer-defined string overflows when a string construction operation is being performed, a help will be issued by the low-level string manipulator: 6d7a6a

ishelp _ HELP(stringoverflow,string : newstring) 6d7a6a1

where STRING is the address of the string. 6d7a6a2

It is expected that a higher level programmer-supplied routine will catch it and resume with the address of a larger string (or extend the existing string if possible). It is also assumed that the catching routine will copy the existing portion of the string to the new area. The routine should: 6d7a6b

RESUME(gothelp,newstring) 6d7a6b1

where NEWSTRING is the address of the new string. 6d7a6b2

The runtime routine will then do 6d7a6c

NOTE(changestring,string,newstring) 6d7a6c1

That should allow all routines that deal with string STRING to change the REF variables that contain the address STRING to NEWSTRING. 6d7a6d

Obviously, one does not want to do this for every string, but just those that are generally of reasonable size but may possibly have to be quite large. 6d7a6e

If no help is obtained for the string manipulator, it does 6d7a6f

ABORT(stringoverflow,$"string too long") 6d7a6f1

Runtime package notes: 7

The interface to the runtime package has been changed. 7a

After loading a program, control should be given to runtime-package label L10START. 7a1

The runtime package will do a procedure call to the procedure whose address is in the variable STARTUP. That variable is in the runtime-package data area and must be setup after loading. 7a2

Instructions for initializing and using an entry vector are given in the front of (nls,x110runtime,). 7a3

In the event of serious runtime error, a runtime-package

procedure called SYSRCV is called to do the recovery operation. SYSRCV+3 is a good debugging breakpoint. The recovery process consists of 7a4

Issuing a NOTE(unwind) to notify the world that it will go away. 7a4a

Resetting stacks and other runtime-package state information. 7a4b

Calling the procedure whose address is in the variable RECOVER. That variable is in the runtime-package data area and must be set up after loading. 7a4c

The RECOVER procedure is called with three arguments: 7a4d

A value which is equal to one of the following pre-defined symbols: 7a4d1

stkoverflow (if error is a runtime-defined stack overflow) 7a4d1a

uncaughtabort (if an ABORT was not dispatched by any catchphrase) 7a4d1b

programbug (if error is one of many that indicate program errors or badly smashed data areas) 7a4d1c

The address of a string describing the error. 7a4d2

An offending address. 7a4d3

When entering the RECOVER procedure the state of the thread of control is exactly as when entering the STARTUP procedure - at ground zero. However, global program variables are unchanged from the time of the error, and the program state with respect to the operating system is also unchanged (e.g. the state of files remains unchanged). 7a4e

It is assumed that debugged programs will not experience a recover situation except under bizarre conditions. 7a4f

A runtime-package procedure named ERRMSG may be called with a file handle, the string address and the error location to have a user-readable string written on the file (location written symbolically). 7a4g

In addition, if the error is an uncaught abort, the signal arguments are saved: 7a4h

syszgn contains the signal name. 7a4h1

syszgt contains the signal type. 7a4h2

syszg2-syszg4 contain agruments 2-4. 7a4h3

Notes on L1011 (PDP-11 version of L10 language): 8

Not Yet Implemented in <SUBSYS>L1011: 8a

List as of 12/8/74 8a1

FIND statement, and related syntax. 8a1a

READC and related syntax. 8a1b

We expect to implement these by 9/75. 8a1c

Programming reminders for PDP-11: 8b

All variables are 16 bits long. 8b1

Strings may be up to 64K bytes, chars are 8 bits long. 8b2

Record fields are limited to 16 bits max length. 8b3

The number of procedure call arguments is limited to 63. 8b4

The number of procedure call results is limited to 4. 8b5

The number of system call arguments/results is limited to 5 8b6

Record fields of length 16, 8 and one bits are significantly faster to reference than other lengths. 8b7

Record field definitions take up significantly more code than on the PDP-10: minimize them. 8b8

String subscript references use hardware addressing and are very fast. 8b9

Although the PDP-11 is a byte address machine, array subscript operations are compatable with PDP-10 programs (i.e. x[3] addresses the third WORD of array x). 8b10

Assmelby code format is similar to PAL-11 with these exceptions: 8b11

The instruction must start with ! (as in PDP-10 L10). 8b11a

No pseudo-ops or macros are implemented. 8b11b

Branch instruction addresses must be written .+n or .-n where n is a compile-time expression that is to be the offset field in the instruction. 8b11c

The address of MARK and SOB instructions must be written Nn where n is a compile-time expression that is to be the low order 6 bits of the instruction. 8b11d

These exceptions are because L10 parses the address of all instructions the same regardless of opcode. 8b11e

DIA 6-FEB-75 08:04 25308

Extensions to the L10 Programming Language for the DEC PDP-10 and DEC PDP-11

Don I. Andrews

6 FEB 75

Augmentation Research Center

Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025

DIA 6-FEB-75 08:04 25308

Extensions to the L10 Programming Language for the DEC PDP-10 and DEC PDP-11

(J25308) 6-FEB-75 08:04;;;; Title: Author(s): Don I. Andrews/DIA;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: POOH; Origin: < META, XL10DOC.NLS;16, >, 5-FEB-75 08:50
POOH ;;;; ####;

DVN 6-FEB-75 09:30 25309

Telephone Log 2/4/75: Continued Interest at Bonneville Power Authority in NLS for Documentation Purposes.

Follows 25216

DVN 6-FEB-75 09:30 25309

Telephone Log 2/4/75: Continued Interest at Bonneville Power Authority in NLS for Documentation Purposes.

Marge Lambie (HJOURNAL, 25216, 1:w) called me late Tuesday afternoon. Since the last journal item she had talked on the phone to Doug and she viewed the movie just before she talked to me. Her interest in NLS seemed considerably more animated. She suggested that Bonneville might go ahead and get a slot for exploratory applications and handle some of their other needs for the present by also gettng some other, more limited system. She was worried about the amount of access on slot provided. I told her that she could be reasonably certain of one or two extra users online after two pacific time. She asked abou multiplexing terminals, and I reminded her about the front-back endsplit planned for NSW. She asked for more information on that and I sent her the NSW proposal. She planned to show the movie to the committee that is associated with finding DPCS service for Bonneville this Thurdsday. 1

DVN 6-FEB-75 09:30 25309

Telephone Log 2/4/75: Continued Interest at Bonneville Power Authority in NLS for Documentation Purposes.

(J25309) 6-FEB-75 09:30;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION] dpcs notebook please) DCE([ACTION]) RLL([INFO-ONLY]) JHB([INFO-ONLY]) PWO([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: DPCS SRI-ARC; Clerk: DVN;

KEV 6-FEB-75 10:35 25310

curren ELF sysgen procedure

This document will describe the procedure I used this week (2/5/75) for putting together an ELF OS. There is no guarrantee that this procedure will be valid in the future as ELF, like the early TENEX, seems to change its sysgen procedure with each new release. 1

BASIC IDEA 2

basically and elf is put together on the AI ten and then ftped to our ten and then loaded. there are a series of runfiles for use here but more on these later. on the AI machine there are two directories that are used: 2a

<ELFDEVEL> 2a1

this directory contains the source files as they are released from SCRL. This is a read-only, files-only directory. 2a1a

<ELF> 2a2

This is a login, working directory. (See me if you need the password.) 2a2a

The basic steps for ELF creation are as follows (more detail below): 2b

1) login at arc 2b1

2) connect to directory elf (password elf) 2b2

3) FTP FROM [SRI-AI]<ELFDEVEL> the most recent version of each of following files: 2b3

ELFCNF.000 2b3a

KERGEN.RUN 2b3b

KERLNK.RUN 2b3c

NCPGEN.RUN 2b3d

NCPLNK.RUN 2b3e

EXECGEN.RUN 2b3f

ELNK.RUN 2b3g

ELFGEN.RUN 2b3h

BINDSYS.RUN 2b3i

4) compare (visually) each of these files with the appropriate old branches in file [SRI-ARC]<ELF>ELF-RUNFILES and update as needed both the old and new branches in [SRI-ARC]<ELF>ELF-RUNFILES 2b4

5) position yourself at the down of each of the new branches and turn on viewspecs 1 (plex only) and B (no indenting) and do an output assembler file to the appropriate nnn.060 files. 2b5

6) FTP from [SRI-ARC]<ELF>*.060 to [SRI-AI]<ELF>*.060 2b6

7) login as elf on the sri-ai machine 2b7

8) run the runfiles in the following order 2b8

ELFGEN.060 2b8a

KERGEN.060 2b8b

NCPGEN.060 2b8c

EXECGEN.060 2b8d

KERLNK.060 2b8e

NCPLNK.060 2b8f

ELNK.060 2b8g

BINDSYS.060 2b8h

9) examine the map and binding files for errors and correct as needed 2b9

10) FTP from [SRI-AI]<ELF>ELFSYS.060 to [SRI-ARC]<ELF>ELFSYS.060 and any other listing or map files you want 2b10

DETAILED PROCEDURE 3

1) login at arc 3a

2) connect to directory elf (password elf) 3b

3) FTP FROM [SRI-AI]<ELFDEVEL> the most recent version of each of following files: 3c

curren ELF sysgen procedure

ELFCNF.000 3c1

this is the configuration file for ELF, 3c1a

KERGEN.RUN 3c2

this is the runfile for compiling the kernel 3c2a

KERLNK.RUN 3c3

this is the runfile for linking the kernel 3c3a

NCPGEN.RUN 3c4

this is the runfile for compiling the ncp 3c4a

NCPLNK.RUN 3c5

this is the runfile for linking the ncp 3c5a

EXECGEN.RUN 3c6

this is the runfile for compiling the exec 3c6a

ELNK.RUN 3c7

this is the runfile for linking the exec 3c7a

ELFGEN.RUN 3c8

this is the runfile for copying the needed files from <ELFDEVEL> to <ELF> 3c8a

BINDSYS.RUN 3c9

this is the runfile for binding together the kernel, the exec, and the ncp 3c9a

4) compare (visually) each of these files with the appropriate old branches in file [SRI-ARC]<ELF>ELF-RUNFILES and update as needed both the old and new branches in [SRI-ARC]<ELF>ELF-RUNFILES 3d

[SRI-ARC]<ELF>ELF-RUNFILES has a bracnh for each of the above files, each branch has two sub-branches: 3d1

an OLD branch that corresponds to the previously released, from SCRL, version of the file, and 3d1a

a NEW branch that reflects the way the file should be for our configuration for the previous release from SCRL 3d1b

This step is the familar merge their updates with our updates step. However this is fairly easy compared to the old TENEX problem since it is mostly runfiles (only exception is the ELFCNF module). 3d2

5) position yourself at the down of each of the new branches and turn on viewspecs 1 (plex only) and B (no indenting) and do an output assembler file to the appropriate nnn.060 files. 3e

6) FTP from [SRI-ARC]<ELF>*.060 to [SRI-AI]<ELF>*.060 3f

7) login as elf on the sri-ai machine 3g

8) run the runfiles in the following order 3h

ELFGEN.060 3h1

KERGEN.060 3h2

NCPGEN.060 3h3

EXECGEN.060 3h4

KERLNK.060 3h5

NCPLNK.060 3h6

ELNK.060 3h7

BINDSYS.060 3h8

9) examine the map and binding files for errors and correct as needed 3i

i have adopted the following extension name naming conventions: 3i1

xxx.SML - macro library source files 3i1a

xxx.K11 - source file for the kernel 3i1b

xxx.N11 - source file for the ncp 3i1c

xxx.E11 - source file for the exec 3i1d

xxx.KL6 - assembly listing files for the kernel 3i1e

xxx.NL6 - assembly listing files for the ncp 3iif

xxx.EL6 - assembly listing files for the exec 3iig

xxx.KO6 - assembly object files for the kernel 3iih

xxx.NO6 - assembly object files for the ncp 3iii

xxx.EL6 - assembly object files for the exec 3iij

K.060 - linked kernel 3iik

N.060 - linked ncp 3iil

E.060 - linked exec 3iim

K.KM6 - link map for the kernel 3iin

N.NM6 - link map for the ncp 3iio

E.EM6 - link map for the exec 3iip

ELFSYS.060 - bound ELF OS 3iiq

10) FTp from [SRI-AI]<ELF>ELFSYS.060 to [SRI-ARC]<ELF>ELFSYS.060 and any other listing or map files you want 3j

KEV 6-FEB-75 10:35 25310

curren ELF sysgen procedure

(J25310) 6-FEB-75 10:35;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: KEV; Origin: < VICTOR, ELF-PROCEDURES,NLS;1, >, 5-FEB-75 18:49 KEV ;;;;#####;

RLL 6-FEB-75 15:48 25311

Franklin phone call of 31 Jan 75

phone contact report - see (25263,) and (25261,)

RLL 6-FEB-75 15:48 25311

Franklin phone call of 31 Jan 75

(DATE) 31 Jan 1975 1

(BY) Lieberman 2

(ATTENDEES) 3

Jeff Franklin (JF5) of NSWC 3a

Robert Lieberman (RLL) of SRI-ARC 3b

(MEDIUM) PHONE 4

(WHERE) Place of contact 5

(ACTION-ITEMS) 6

Actions taken, to be taken, etc., dated 6a

(DISTRIBUTION) DCE JCN RLL 7

(REFERENCES) (25261, 1:w) Visit by Jeff Franklin of NSWC 17 Jan 1975 8

(REMARKS) 9

Franklin called today to report on what happened at the meeting with OSHA people. 9a

He reported very positive response to our system from Boyd and others at the meeting. They (OSHA, Boyd) plan to call us to set up a demonstration for them in Washington area. 9b

Jeff will keep posted on what happens but will not be actively involved as a go-between. 9c

(ADDRESSES) Full name of organization, address, and phone number 10

(DOCUMENTS) Hard copy given and received 11

(GIVEN) Date and documents given 11a

(RECEIVED) Date and documents received 11b

RLL 6-FEB-75 15:48 25311

Franklin phone call of 31 Jan 75

(J25311) 6-FEB-75 15:48;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /DCE([INFO-ONLY]) JCN([INFO-ONLY]) RLL([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

CHI 7-FEB-75 11:54 25312

Meeting Monday

There will be a meeting at 1:30 Monday (10-feb) to discuss the recent nsw review meeting attended by rww and jbp and to discuss issues related to our transfer to BBN's TENEX and our PDP-11s. 1

CHI 7-FEB-75 11:54 25312

Meeting Monday

(J25312) 7-FEB-75 11:54;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: CHI;

The Most Useless Command Contest

We are accepting entries for this contest through the end of next week. All entires must be twenty-five words or less, but you may enter as many times as you wish. Please include your return address and note that this offer is void where prohibited. Several entries have already been received and we will publish these to give you examples:
Dirk: Show Herald
Kirk: Force Case Invisible (with filter)
Pooh: Force Case Number
There will be a prize for the winner and a contest for the folloing week has already been planned.

1

The Most Useless Command Contest

(J25313) 7-FEB-75 15:13;;;; Title: Author(s): Dirk H. Van Nouhuys, Ann Weinberg/DVN POOH; Distribution: /SRI-ARC([ACTION]) DLS([ACTION]) GCE([ACTION]) DHC([ACTION]) GSG([ACTION]) NJN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: POOH;

PI Meeting

7-JAN-75 1333-PST REID at USC-ISIB: DRAFT LETTER OF INVITATION TO PI'S
Distribution: BLUE AT ISI, feinler at sri-arc, uncapher
Received at: 7-JAN-75 14:42:05 1

DRAFT 1a

MEMORANDUM FOR PRINCIPAL INVESTIGATORS 1b

THIS IS YOUR PERSONAL INVITATION TO ATTEND THE 1975 ARPA/IPT PRINCIPAL 1c

INVESTIGATORS CONFERENCE, WHICH WILL BE HELD IN SAN DIEGO, CALIFORNIA. 1d

THE FIRST SESSION WILL CONVENE AT 9:00 A.M.ON WEDNESDAY, MARCH 12, AND 1e

THE FINAL SESSION WILL CONCLUDE AT 4:00 P.M. ON FRIDAY, MARCH 14, THE 1f

CONFERENCE WILL BE HELD AT THE TRAVELODGE HARBOR ISLAND, SAN DIEGO, 1g

CALIFORNIA. EACH PRINCIPAL INVESTIGATOR SHOULD MAKE HIS OWN HOTEL 1h

ROOM RESERVATION VIA THE ENCLOSED RESERVATION CARD TO ASSURE GROUP 1i

SINGLE RATE OF $19.00/DAY. IF YOU WISH TO CONTACT THE HOTEL DIRECTLY, 1j

THE NUMBER IS (714) 291-6700 AND IDENTIFY YOURSELF AS ARPA. PLEASE 1k

MAKE HOTEL RESERVATIONS BY FEBRUARY 25. FREE LIMOUSINE SERVICE TO THE 1l

HOTEL IS AVAILABLE BY USING THE FREE PHONE IN THE BAGGAGE CLAIM AREA 1m

AT THE SAN DIEGO AIRPORT. THE HOTEL IS 1/3 MILE FROM THE AIRPORT. 1n

THE CONFERENCE WILL BE STRUCTURED ALONG THE SAME GENERAL LINES AS IN 1o

THE PAST THREE YEARS, EACH PRINCIPAL INVESTIGATOR IS REQUESTED TO 1p

PI Meeting

SUBMIT TO ELIZABETH (JAKE) FEINLER (SRI) VIA THE NET, A WRITTEN 2-PAGE 1q

SUMMARY OF HIS PROJECT ACTIVITIES DURING THE CALENDAR YEAR 1974. A 1r

SAMPLE TWO-PAGE REPORT IS ATTACHED. THE WRITTEN DESCRIPTION SHOULD 1s

NOT EXCEED 2 PAGES IN LENGTH. INSTRUCTIONS FOR SUBMITTING THE REPORT 1t

VIA THE NET ARE ATTACHED. 1u

2 1v

FOR THOSE NOT HAVING ACCESS TO THE NET, PLEASE SEND ONE COPY OF YOUR 1w

REPORT TO: 1x

MISS MAGGIE REID 1y

UNIVERSITY OF SOUTHERN CALIFORNIA 1z

INFORMATION SCIENCES INSTITUTE 1a@

4676 ADMIRALTY WAY 1aa

MARINA DEL REY, CALIFORNIA 98291 1ab

(213) 822-1511 1ac

THE MATERIAL MUST BE IN EITHER JAKE FEINLER'S OR MAGGIE REID'S HANDS 1ad

BY FEBRUARY 20. THE REPORTS WILL BE DUPLICATED AND INSERTED IN THE 1ae

NOTEBOOKS FOR DISTRIBUTION AT THE BEGINNING OF THE CONFERENCE. 1af

(END) 1ag

NOTE - AL BLUE: THE FOLLOWING NEEDS TO BE DONE AT IPTO PRIOR TO 1ah

SENDING THE LETTER: A) ASSIGNMENT OF UNIQUE NIC NUMBERS; B) DECISION 1ai

ON THE CONTENT OF THE MEETING, INCLUDING THE POSSIBILITY OF TOURS 1aj

(PERHAPS) AS PART OF THE THREE DAY SCHEDULE (RUSSELL SUTHERLAND); C) 1ak

SELECTION OF A MODEL 2-PAGE REPORT FOR INCLUSION; D) ATTACH THE HOTEL 1al

RESERVATION FORMS AND THE INFORMATION CARD; E) ATTACH TO EACH LETTER: 1am

1. INSTRUCTIONS FOR SUBMITTING REPORTS 2. SAMPLE HEADING 3. SAMPLE 1an

REPORT. 1ao

3 1ap

INSTRUCTIONS FOR SUBMITTING REPORTS 1aq

PLEASE USE THE HEADING FORM ENCLOSED, AND PUT THE NIC NUMBERS (TO BE 1ar

ASSIGNED BY IPTO) ON THE SECOND PAGE ALSO. SEND YOUR 2-PAGE SUMMARY 1as

TO SRI-ARC VIA FTP GIVING THE PAPER THE FILE NAME: 1at

<PI>LASTNAME.TXT 1au

LOG IN AS user=NIC-WORK, password=ARPA, then use FTP. 1av

4 1aw

NIC (UNIQUE TO A PI) 1ax

PART OF NIC 24980 1ay

(TITLE) 1az

1974 IPTO PROJECT SUMMARY 1b@

PREPARED FOR: ARPA IPT PRINCIPAL INVESTIGATORS MEETING 1ba

MARCH 12-14, 1975 1bb

PI Meeting

PREPARED BY: PRINCIPAL INVESTIGATOR'S NAME 1bc

FULL ADDRESS 1bd

(PLEASE PUT YOUR NIC NUMBER ON SECOND PAGE ALSO) 1be

KEITH 1bf

P.S. PLEASE ACCEPT MY APOLOGIES FOR THE "DRAFT" DRAFT!
THANKS, MAGGIE REID 1bg

REID FOR UNCAPHER 1bh

JAKE 7-FEB-75 15:27 25314

PI Meeting

(J25314) 7-FEB-75 15:27;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /DCE([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

Informal Documentation Report for Week ending 2/8

POOH 1

glossary: continued revisions that recylcled back through Dirk and Kirk. Selected examples that were chosen with the help of Susan are still to be added. 1a

business cards were received from COM and sent through to be printed. SRI red tape is holding up that pocess. 1b

journalized and printed Don Andrews' new L10 document. 1c

Complained a lot about the system being overloaded. 1d

KIRK 2

> Wrote a beautiful scenario describing what sending mail would be like in the NSW only to have it deleted by my keyset changing text into plex. 2a

> updated <documentation, helpd,> from hgl and pooh's comments. 2b

> Wrote a general description of NSW help description files in <micheal, janscen,> to be journalized. 2c

> Reviewed POOH's help work. 2d

> started on the new glossary generation program. 2e

DVN 3

NSW Documentation 3a

Issued a new work allocation including tenative schedules. POOH, KIRK and I agreed to meet Monday at 1:30 to get together on plans, structure, and procedures for the new Help files we will begin to generate. 3a1

Help/Glossary 3b

revision of Help is finished except fr a few TNLS examples and Anne's reading in edits by KIRK and I on the tial ed of her work. We are starting wrk on the hardcopy production of the glossary. 3b1

Final Report 3c

Dick and I have brought all our writing atleast to the draft stage. Charles and Harvey still owe writing. I have begun integration work. 3c1

Informal Documentation Report for Week ending 2/8

Small Trailing NLS-8 Documents 3d

Preface to NLS: Waiting for Application's Review 3d1

TNLS Addressing: It is on me to repsond to RWW's review. 3d2

COM: 3e

The revised command summary awaits my attention for COM printing. 3e1

The TNLS-8 Primer awaits my attention for COM printing. 3e2

Marin Hardy's paper Microprocessing Technology awaits my attention for small revisons befor final COM run. 3e3

COM version of Ken Victor's CML Paper returned from SRI prining. It looks good. I asked Joan to send a few copies to Chuck Dornbush since he appeares as first author. 3e4

Began Discussions with Jake Feinler about COMing the next Resource Notebook. POOH willwork with me to learn more about COMing. 3e5

POOH KIRK DVN 7-FEB-75 16:48 25315

Informal Documentation Report for Week ending 2/8

(J25315) 7-FEB-75 16:48;;;; Title: Author(s): Ann Weinberg, Kirk E. Kelley, Dirk H. Van Nouhuys/POOH KIRK DVN; Distribution: /JOAN([ACTION] dirt notebook please) DIRT([INFO-ONLY]) ; Sub-Collections: DIRT SRI-ARC; Clerk: DVN;

CHI 10-FEB-75 09:53 25318

NSW -- Machine transfer meeting rescheduled

The meeting anounced in 25312 will be at 1:00 instead of 1:30. 1

CHI 10-FEB-75 09:53 25318

NSW -- machine transfer meeting rescheduled

(J25318) 10-FEB-75 09:53;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: CHI;

DVN 10-FEB-75 10:35 25319

Cost for Hard Copy Printing after COM

Frank Brignoli asked for this information; I am sending this item to several other people I thought might be interested. 23549, 24406, and 24105 contain related information.

DVN 10-FEB-75 10:35 25319

Cost for Hard Copy Printing after COM

After you add quality control, overhead, etc., printing at SRI costs about $.75 a page for ITEK Masters (good for about 1000 copies) and about 2 cents per impression (impression = copies X pages) for printing in the range of 100-500 copies. 1

Cost per impression goes down as you make more copies. Other possible costs are $1-5 per page that includes line drawing or halftone illustrations, and various costs for various types of covers and binding. Typical inexpensive cover and binding is $.45 per copy. 2

That means 100 copies of a typical 100-page report without artwork but with covers would cost about $225.00. 3

I will mail you a sheet with more detailed costs. 4

Offset printing is a genuine open marketplace. That means costs vary with geographical location, whom you talk with, what he thinks he can stick you for, and the leverage and skill of the person who negotiates with the vendor. In general SRI, for accounting reasons, is a moderately expensive place to print. In general the Bay Area is high cost. I suppose NSRD has a large enough volume to have some leverage with local printers and has some one who is experienced in negotiating with them. Maybe not if you are required to go through the Government Printing Office. 5

For the future. As you may recall we are negotiating with George Lithograph for COM service (link). Unlike DDSI, George is a printer and wants to do the printing themselves. My guess is their prices will be moderately lower than SRI's. 6

DVN 10-FEB-75 10:35 25319

Cost for Hard Copy Printing after COM

(J25319) 10-FEB-75 10:35;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /FGB([ACTION] Did you ever get samples of the JOVIAL Manual form Duane Stone?) JOAN([ACTION] dpcs notebook please) DPCS([INFO-ONLY]) VGK([INFO-ONLY]) WEC([INFO-ONLY]) LAC([INFO-ONLY]) EAR([INFO-ONLY]) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN;

JCN RA3Y 9-FEB-75 13:00 25320

Office-1 User and System Accounts and Allocations

Office-1 group allocations are being changed to conform to the pending contracts for service today 2/9. Most allocations have been in line with the coming contracts during the past few weeks. In addition, user account numbers are being changed to provide bette use data by subscriber . Users may login without knowing their new number by typing altmode or escape (to see the number) or CR. If any other than the default number is typed, users will get an error message advising then to start over and to use altmode or escape at the account number request point. It should work. Jim N.

Office-1 User and System Accounts and Allocations

Group	Accounts New	Accounts Old	Allocations New	Allocations Old	Alloc Group # New	Alloc Group # Old	
System	10	0,1,10	-	-	special	special	
Tymshare	20	20	-	-	special	special	
ARC-UTIL	30	30	-	-	special	special	
Consultants special		90	-	-	-	special	
NSRDC	200	46	2	1	-	13	
HUDSON	320	80	1	1	-	16	
ETS		340	80	1	1	-	14
ARC-APP	360	-	1	0	-	-	
ARC-MGT	380	-	1	0	-	-	
RADC	400	40	5	5	-	5	
AFAA	440	-	-	0	with RADC	-	
BELL	500	50	1	1	-	4	
BRL	600	45	1	1	-	12	
SRI	700	35	1	1	-	10	
ARPA	800	80	5	6	-	7	
MIT-SEISMIC		820	80	2	1	-	9
NICGUEST	840	3	-	1	with ARPA	6	
ARPA-NSW	880	80	3	0	-	-	
NSA	900	90	1	1	-	15	

Office-1 User and System Accounts and Allocations

Accounts by Directory:

Group	New	Old
OVERHEAD USERS		
System	10	0,1,10

CAT:10*
CAT-PROGS:10*
ACCOUNTS:10*
AJOURNAL:10*
ARCHIVE:10*
BACKGROUND:10*
BJOURNAL:10*
BSYS:10*
CATALOG:10*
CJOURNAL:10*
DIAGNOSTICS:10*
DJOURNAL:10*
DOCUMENTATION:10*
DUVALL:10*
EJOURNAL:10*
EXEC:10*
FJOURNAL:10*
GJOURNAL:10*
HJOURNAL:10*
IDENTFILE:10*
IJOURNAL:10*
IMLAC:10*
JJOURNAL:10*
JOURNAL:10*
KJOURNAL:10*
LJOURNAL:10*
MJOURNAL:10*
NET:10*
NETPROG:10*
NETSYS:10*
NIC-NLS:10*
NLS:10*
OUTJOURNAL:10*
PMFDIR0:10*
PRINTER:10*
REL-NLS:10*
SOURCES:10*
SRIACCT:10*
SUBSYS:10*
SYSTEM:1*,10
TEJOURNAL:10*
TENEX:10*
TIPUG:10*

Office-1 User and System Accounts and Allocations

USER-PROGS:10*
USERGUIDES:10*

Office-1 User and System Accounts and Allocations

```
Tymshare     20                20
   MARTINEZ:20*
   NEUMANNR:20*
   OPER:20*
   POLLACK:20*
   ROY:20*
   SANFILIPPO:20*
   WHEAT:20*
```

Office-1 User and System Accounts and Allocations

ARC Utility	30	30
BAIR:30*		
BECK:30*		
FEEDBACK:30*		
HARDWARE:30*		
HARDY:30*		
HOPPER:30*		
JOHNSON:30*		
JORDAN:30*		
KELLEY:30*		
LEE:30*		
MEYER:30*		
NORTON:30*		
PETERS:30*		
ROETTER:30*		
VICTOR:30*		
WHITE:30*		

Office-1 User and System Accounts and Allocations

```
Consultants          90              ?
   BBN-NET:90*
   BBN-ORG:90*
   BBN-TENEX:90*
   BTHOMAS:90*
   CLEMENTS:90*
```

Office-1 User and System Accounts and Allocations

SUBSCRIBING* USERS

[Reserved 100 -] next new orgs: 300 series

NSRDC 200 46

AVRUNIN:200*
BRIGNOLI:200*
COMRADE:200*
ISDS:200*
MATHSCI:200*
NALCON:200*
NAVAPS:200*
NAVIMP:200*
NAVINFO:200*
NAVLIS:200*
NAVMINI:200*
NAVSEC:200*
NSRDC:200*

Office-1 User and System Accounts and Allocations

```
HUDSON      320                 80
   GIACOBINC:320*
   ROHRBAUGH:320*
   RUGGLES:320*
```

Office-1 User and System Accounts and Allocations

```
ETS                     340                 80
   ANASTASIO:340*
   MCNALLY:340*
   POTTER:340*
   RUMAR:340*
   TRYOUT:340*
   VANHASSEL:340*
```

Office-1 User and System Accounts and Allocations

```
ARC-APP    360
   LIEBERMAN:360*
   NETINFO:360*
   NIC:360*
   PANKO:360*
   RATNER:360*
   VANNOUHUYS:360*
```

Office-1 User and System Accounts and Allocations

```
ARC-MGT     380
   ENGELBART:380*
   LEAVITT:380*
   WATSON:380*
```

```
RADC        400                 40
    BARNUM:400*
    BERGSTROM:400*
    BUCCIERO:400*
    CALICCHIA:400*
    CARRIER:400*
    CAVANO:400*
    DAUGHTRY:400*
    DECONDE:400*
    DIMAGGIO:400*
    FEMIA:400*
    HILBING:400*
    IUORNO:400*
    KENNEDY:400*
    KENYON:400*
    KESSELMAN:400*
    KRUTZ:400*
    LAFORGE:400*
    LAMONICA:400*
    LAWRENCE:400*
    LIUZZI:400*
    LOMBARDO:400*
    LORETO:400*
    MCLEAN:400*
    MCNAMARA:400*
    NELSON:400*
    PANARA:400*
    PATTERSON:400*
    PETELL:400*
    RADC:400*
    RWALKER:400*
    RZEPKA:400*
    SLIWA:400*
    STELLATO:400*
    STINSON:400*
    STONE:400*
    THAYER:400*
    TOMAINI:400*
    VANALSTINE:400*
    WEBER:400*
    WINGFIELD:400*
    WWMCCS:400*
```

Office-1 User and System Accounts and Allocations

AFAA 440 -

Office-1 User and System Accounts and Allocations

```
BELL        500                 50
    ATKINSON:500*
    BEDFORD:500*
    BELL:500*
    DAY:500*
    DDAY:500*
    FELDMAN:500*
    GEDWARDS:500*
    HOYLE:500*
    KATSOULIS:500*
    KOLLEN:500*
    MATTIUZ:500*
    MEADE:500*
    NAPKE:500*
    VU:500*
    WEINTRAUB:500*
```

```
BRL         600                 45
    ARNTSON:600*
    AYERS:600*
    BRL:600*
    CIANFLONE:600*
    CUMMINGS:600*
    DSMITH:600*
    DTAYLOR:600*
    GILBERT:600*
    HARRISON:600*
    LEISHER:600*
    MITCHELL:600*
    PROBERTS:600*
    PULLEN:600*
    TAYLOR:600*
    UHLIG:600*
    WRUBLEWSKI:600*
```

Office-1 User and System Accounts and Allocations

```
SRI          700                35
   BERTRAND:700*
   ELLIOT:700*
   GREEHAN:700*
   GRIMM:700*
   HOUGH:700*
   HUMPHREY:700*
   MABREY:700*
   O'KEEFE:700*
   PLACKO:700*
   PORT:700*
   RIPPLE:700*
   SRI-TRAINEE:700*
```

```
ARPA        800                 80
    ARPA-PM:800*
    ARPA-PRACTICE:800*
    ARPA:800*
    BANGERT:800*
    BARNES:800*
    BEARD:800*
    BECKER:800*
    BLACK:800*
    BLUE:800*
    CAMPBELL:800*
    CARLSON:800*
    CARLSTROM:800*
    CERL:800*
    CHAPMAN:800*
    COLEMAN:800*
    COLEMAN:800*
    COOK:800*
    CROCKER:800*
    DCLEMENTS:800*
    DORIS:800*
    DUBOIS:800*
    EDWARDS:800*
    FAVOR:800*
    FEDERHEN:800*
    FEDERHEN:800*
    FIELDS:800*
    FLO:800*
    GLAWRENCE:800*
    GOERING:800*
    HARRIS:800*
    HARTSELL:800*
    HEILMEIER:800*
    HELGA:800*
    HILDA:800*
    HYDE:800*
    IANSON:800*
    IWWSS:800*
    JACKSON:800*
    JALLEN:800*
    JOAN:800*
    JONES:800*
    JTSA-O:800*
    KAHN:800*
    KALLAS:800*
    KIBLER:800*
    KING:800*
    KIRKWOOD:800*
    KOBLISKI:800*
```

```
KRESA:800*
LICKLIDER:800*
LUDWIG:800*
LUKASIK:800*
LYONS:800*
MCLINDON:800*
MSTONE:800*
NIEDENFUHR:800*
ORSINI:800*
PARISI:800*
PCLARK:800*
RMOORE:800*
ROMNEY:800*
ROWENA:800*
RUBY:800*
RUSSELL:800*
RYOUNG:800*
SDPCC:800*
STALOG:800*
STICKLEY:800*
STO:800*
STUBBS:800*
SULLIVAN:800*
TACH:800*
TAO:800*
TTO:800*
VANDERBURGH:800*
VANREUTH:800*
WALKER:800*
WALSH:800*
WILKINS:800*
WILLIS:800*
WORCH:800*
WORCH:800*
XGP:800*
YEE:800*
ZIEBELL:800*
```

```
MIT-SEISMIC          820                80
   CCA:820*,3
   DOCB:820*
   LACOSS:820*
   SDAC-TIP:820*
   SHEPPARD:820*
   SWIM:820*
```

Office-1 User and System Accounts and Allocations

NICGUEST 840 3
NICGUEST:840*

Office-1 User and System Accounts and Allocations

```
ARPA-NSW   880                80
   CRAIN:880*
   CROFT:880*
   FALLEN:880*
   FINNEY:880*
   FRALICK:880*
   JACOBS:880*
   KEHLER:880*
   LLOYD:880*
   LUTKENHOUSE:880*
   MAHLUM:880*
   MAHONEY:880*
   MOONEY:880*
   MORTENSON:880*
   RIDDLE:880*
   SLEZYCKI:880*
   WEEKS:880*
```

Office-1 User and System Accounts and Allocations

```
NSA          900                90
    BAILEY:900*
    HASSING:900*
    HELP:900*
    HILL:900*
    MADDEN:900*
    MATHESON:900*
    MCCLOGHRIE:900*
    MITRE-TIP:900*
    MUMAUGH:900*
    NOGA:900*
    NSA:900*
    ROBERTAZZI:900*
    ROCHE:900*
    TAGGART:900*
```

to be deleted:
BROWN:60*
CAPPS:60*
ENERGY:60*
KERNS:70*
KRUZIC:70*
MILLER:70*
RODDEN:70*
WALTERS:70*
WHITBY:70*

JCN RA3Y 9-FEB-75 13:00 25320

Office-1 User and System Accounts and Allocations

(J25320) 9-FEB-75 13:00;;;; Title: Author(s): James C. Norton, Raymond R. Panko/JCN RA3Y; Distribution: /JCP([ACTION]) KWAC([INFO-ONLY]) ; Sub-Collections: SRI-ARC KWAC; Clerk: JCN; Origin: < NORTON, NEWALLOCATIONS.NLS;1, >, 9-FEB-75 12:20 JCN ;;;; ####;

JHB 10-FEB-75 12:43 25321

What To Do About Commands Not Implemted Or Prohibitively Bugged , re. 31806,

Suggestion in response to POOH's question.

JHB 10-FEB-75 12:43 25321

What To Do About Commands Not Implemted Or Prohibitively Bugged , re. 31806,

If the command exists so that a user sees it with the question mark facility or documentation (online or off), then it should be documented, with a statement about its current condition. Thus, if it were repaired it would be documented and a minor editing change could indicate the new status. 1

During training or user assistance there should be no mention of things we know do not work,...and no promise that they will. 2

Ideally, these commands would be "commented out" of the CML so that a user would not see them, but currently there are no resources for this. If any of us finds out that a particular thing has been fixed, he should let all of us know about it (the ident UD for User Developmentis a good distribution). 3

JHB 10-FEB-75 12:43 25321

What To Do About Commands Not Implemted Or Prohibitively Bugged , re. 31806,

(J25321) 10-FEB-75 12:43;;;; Title: Author(s): James H. Bair/JHB; Distribution: /UD([ACTION]) POOH([ACTION]) KIRK([ACTION]) DVN([ACTION]) RLL([ACTION]) RA3Y([ACTION]) MEH([ACTION]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC UD; Clerk: JHB;