DIA 21-JUL-71 13:50 7415

response memo

Response memo 1

We are now running with these improvements 1a

NLS is dismissing itself for 50 ms. on completion of command execution. This is supposed to reduce the reserved memory for the fork but it has not been extablished that thatactually happens. 1a1

NLS is now mapping 96 file pages so that even very big files will be kept on the drum. Jumping around inside large files should now be faster, once the pages hve been moved from the disk to the drum. 1a2

The statistics program now prints the average time that programs wait on the go list. 1a3

The monitor is now removing from the balance set any forks that fault on a page that has to come from the disk. This frees up memory. 1a4

The number of runnable jobs in the balance set has increased due to the above change. 1a5

Previously, about 60 to 70% of the jobs in the BS were in page wait. 1a5a

Now only about 50% are in page wait. 1a5b

The statistics are now correct with any drum configuration. 1a6

The elevator algorthm is now in, and doing very well. Average disk transfer time went down from about 200+ ms. to 125 ms. 1a7

These changes will be in the system soon: 1b

Ken is going to try a minimum disk head movement altorthm. 1b1

A new set of queue times will hopefully give better response by descriminating between interactive and computebound jobs better. This will mean poorer response for very long NLS commands, however. 1b2

A few parameters will be changed: 1b3

PTAV ← 20 ms. 1b3a

BTFACT ← 10ms/page (instead of 20) 1b3b

Forks will not be charged for page faults 1b3c

Some people are going to run NLS use measurements regular for awhile. 1b4

NEWST has been rewritten to put forks dismissed for teletype or workstation input on queue zero and give others some credit for waiting. 1b5

This makes t much simpler. Before it was ignoring forks that waited for less than 100ms and it gave others a priority equal to queue zero for one queue zero quantum. 1b5a

Plans: 1c

NLS's working set should be smaller, so that we can get more jobs in the balance set (primarily to reduce I/O wait time) 1c1

If this cannot be done within NLS, the system parameter for max working set size can be changed. 1c1a

The teletype input stuff will help reduce the memory squeeze since the display JSYS code can then be swappable. 1c2

A SIN for displays needs to be invented, but it should be compatable for IMLACS over the NETWORK. 1c3

DIA 21-JUL-71 13:50 7415

response memo

(J7415) 21-JUL-71 13:50; (Expedite) Title: Author(s): Don I. Andrews/DIA; Distribution: William H. Paxton, John T. Melvin, Charles H. Irby, Roger D. Bates, Richard W. Watson, Bruce L. Parsley, Don C. Wallace, Mimi S. Church/WHP JTM CHI RDB RWW BLP DCW MSC; Keywords: response; Sub-Collections: ARC; Clerk: DIA;
Origin: <ANDREWS>MEMO.NLS;1, 21-JUL-71 13:44 DIA ;

Requirements for a Stage 1 NIC Online Resource System 1

There are a number of bodies of information, functional documents, which the NIC needs to maintain online in a fashion easy to access and use from typewriter terminals. The Stage 1 system will utilize present viewing and content searching capabilities of NLS. 1a

The initial documents to be online with the initials of those responsible for their maintenence are: 1b

(1) The TNLS Guide (MFA) 1b1

(2) The Network Resource Notebook (BBN JBN) 1b2

(3) The Current Catalog of the NIC Collection (JBN) 1b3

(4) The Directory of Network Participants (JBN) 1b4

(5) Site Status Information (yet to be created by BBN or ourselves) 1b5

(6) A guide on How to Use the Online Resource System (DVN) 1b6

There needs to be a person in charge of maintaining the set of contents filesd in a current state. 1c

The system will be a three-level system. 1d

Level 1 will be a master contents file with links to the NIC functional contents files of Level 2. 1d1

Level 2 will contain detailed contents files of the functional documents with links to actual content documents in the Journal. 1d2

Level 3 will be the subdocuments of each functional document which will reside in the Journal. 1d3

The material in the contents will be arranged to make effective use of the level and line clipping capabilities of NLS. Entry to an actual document will be with statement numbers on, so that section heading can serve as a further direction to the search. The statement numbers will help the typewriter users. 1e

The Table of Contents of a functional document will also make use of one line citations, levels, links for increasingly more

precise initial entry points in the actual documents. For example, in the Catalog of the NIC Collection the author index contents could contain finer contents with links to the head of each alphabetic section, the A's, B's, etc. Or we could segment the material and index it with citations such as are used at the top of a dictionary page . 1f

With such well-designed contents sections and links, scanning an online document with a typewriter should be feasible, particularly when coupled with content addressing, where appropriate. 1g

RWW 21-JUL-71 13:58 7416

Requirements for a Stage 1 NIC ONline Resource System

(J7416) 21-JUL-71 13:58; (Expedite) Title: Author(s): Richard W. Watson/RWW; Distribution: James C. Norton, Jeanne B. North, Dirk H. van Nouhuys, Bruce L. Parsley, Marilyn F. Auerbach, William S. Duvall/JCN JBN DVN BLP MFA WSD; Sub-Collections: ARC; Clerk: RWW;
Origin: <WATSON>RESOURCE.NLS;3, 21-JUL-71 13:54 RWW ;

Line Printer Purchase Rationale

Line Printer Purchase Rationale 1

Line printer was offered to us by Data Products Corp. for $22,645. The price was based on a total price of $38,600, less 100 per cent of the rental paid thus far. 1a

We rejected this offer in favor of continued leasing because we do not plan to use the printer long enough to justify the purchase. 1b

We shall consider purchasing the printer again after our study of other hard copy output equipment is completed and pertinent decisions have been made. 1c

Ed VanDeRiet 1d

EKV 21-JUL-71 16:48 7418

Line Printer Purchase Rationale

(J7418) 21-JUL-71 16:48; Title: Author(s): Ed K. Van De Riet/EKV; Distribution: James C. Norton, Barbara E. Row/JCN BER; Keywords: ; Sub-Collections: ARC; Clerk: BER;

SCHEDULER GLOSS 1

States of jobs 1a

Every fork in the system is in one of the following states: 1a1

On the wait list. 1a1a

It has associated with it a test routine and data word. The routine will tell the system if the wait condition is satisfied. 1a1a1

(in FKSTAT RH and LH respectively) 1a1a1a

Also has the clock reading at the time it was put into a waiting state so that waiting time can be computed. 1a1a2

(in FKPGST) 1a1a2a

On the go list. 1a1b

<SRI-MOD> Associated with it is the time it was put on the go list. 1a1b1

(in FKPGST) 1a1b1a

In the balance set. 1a1c

Forks in the balance set are either 1a1c1

running 1a1c1a

runnable 1a1c1b

in a page wait state (waiting for a page that it faulted on) 1a1c1c

(in FKPGST) 1a1c1c1

or designated for removal but not really removed because it is waiting for a page. Fork will be put on the go list when the page gets in. These forks are not really in the balance set so far as taking up memory is concerned. 1a1c1d

Associated with each fork in the system: 1b

Queue number. 1b1

There are five queues in the system (0 - 4). Every fork in the system has a queue number. 1b1a

Queue quantum. 1b2

With each of the queues there is a time or quantum. 1b2a

(QBASE[q] - QBASE[q+1]) 1b2a1

When a fork runs on a queue for more than that quantum, he is put on the next queue. Forks on the last queue are put on the "front" of the last queue again on quantum overflow. 1b2b

Time on queue. 1b3

When a fork is put on a queue, the clock reading is recorded so that the real time on that queue can be computed. 1b3a

(in FKTIME) 1b3b

Working set parameters. 1b4

Every fork has: 1b4a

WS, number of pages in memory (working set). 1b4a1

(in FKWSP.RH) 1b4a1a

NR, system's estimate of the number of pages in WS when that fork is running. 1b4a2

(in FKNR.RH) 1b4a2a

Tav, the average time in ms. between page faults for that fork. 1b4a3

(in FKWSP.LH) 1b4a3a

The priority scheme. 1c

The only thing that makes the queues really behave at like queues is the priority scheme. 1c1

Basically, every fork has a priority that is constantly changing in real time. While on a given queue, a forks priority increases in real time at a rate depending on several parameters, but independent of how much it runs. 1c2

Actually, the highest priority is represented by zero, and the larger the priority number, the smaller the priority. 1c3

(computed by CORFCT) 1c3a

The initial priority when placed on a specified queue and the rate of increase are determined by the following: 1c4

TBASE[q] contains the initial priority, i.e. TBASE[q] is the priority on queue q for zero ms. 1c4a

The priority number decreases in time to zero (high priority) at the rate of 2↑f[q] where f[q] is zero or negative. 1c4b

(f is actually called TFACTR) 1c4b1

That is, the priority is determined by 1c4c

TBASE[q] - t*2↑f[q] 1c4c1

The higher the queue number, the higher the initial priority number (lower the priority) and also, the slower the rate at which that priority number decreases (priority increases). 1c4d

But it is possible to have a compute bound job with a priority higher than a job just put on queue zero -- which was not possible on the 940 due to strict queueing. 1c4e

The scheduler loop. 1d

The following pseudo-program contains the essence of the scheduler loop. 1d1

(SCHED0) 1d1a

Execute scheduler clocks if exhausted 1d1a1

qntdms - quantum overflow dismis 1d1a1a

skdlv8 - imp and big tty buffer service 1d1a1b

clk2 - device check for disk, tape, punch, etc. 1d1a1c

Execute scheduler requests if any 1d1a2

(start job) 1d1a2a

If ISKED set, call SCHED1 to test all waiting jobs 1d1a3

If SKEDF2 or PSKED set, call DISMSJ to dismis current fork 1d1a4

SKEDF2 is scheduler flag 1d1a4a

PSKED means page read completed 1d1a4b

If switches request, do it by calling SWTST 1d1a5

If there is no current fork, call SKDJOB to get one 1d1a6

Continue running fork 1d1a7

(SCHED1) 1d1b

For all waiting jobs 1d1b1

call test routine with data 1d1b1a

skip return: call NEWST 1d1b1a1

NEWST uses waiting time to recompute queue level, quantum. 1d1b1a1a

Set SKEDF2. 1d1b1a1b

otherwise continue loop 1d1b1a2

Return 1d1b1b

(SKDJOB) 1d1c

Do a core garbage collection if necessary 1d1c1

Is memory overcommitted (sum of reserved > max) or number of balance set jobs over max? 1d1c2

If so, remove a job by calling REMJOB 1d1c2a

REMJOB picks the job with MAX priority number from CORFCT, but does not remove anyone who has been running in balance set less than WS*BTFACT ms. 1d1c2a1

Is there a job just entering the balance set, or no job waiting to enter? 1d1c3

If NOT, load a job by calling LDJOB and GOTO SKDJOB again 1d1c3a

(SKDJ7) 1d1c4

Reset PSKED (to detect recent page read completes) 1d1c4a

Select the best runnable job in BS, i.e. minimum priority number from CORFCT. 1d1c4b

At the same time, test jobs in page wait to see if their page has come in. Remove jobs that are designated for removal and their page has come in. 1d1c4b1

At this point, if any page transfers have finished, GOTO SKDJ7 1d1c4b2

Setup to run selected process 1d1c4c

RETURN 1d1c4d

(LDJOB) (not really a procedure since it does not return) 1d1d

Select the best of ready jobs by calling SCDRUN 1d1d1

SCDRUN will call CORFCT to find job on golist with MIN priority number 1d1d1a

load the fork into the balance set if 1d1d2

it is better (priority number lower than) the worst job in balance set 1d1d2a

or there is room for one more job in balance set 1d1d2b

and the sum of reserved pages will fit 1d1d2b1

and there are enough free pages to load PSB and UPT for him 1d1d2b2

If you loaded a job, GOTO SKDJOB 1d1d3

Otherwise GOTO SKDJ7 1d1d4

How the scheduler is invoked: 1e

The channel 7 interrupt is used to enter the scheduler. The scheduler is not always entered however: 1e1

If ISKED is set, or if a 60Hz clock interrupt has occurred, the scheduler clocks are updated. Then, if ISKED is set (and it will be if any clocks ran out or anyone else set it), the scheduler is then entered. 1e1a

The channel 7 interrupt is triggered by: 1e2

The 60Hz clock interrupt 1e2a

A page read complete (but ISKED is not set) 1e2b

Any monitor routine that wants to do accurate timming (such as RUNTIM JSYS from BBN) just for the purpose of updating clocks. 1e2c

A few other places in the monitor. 1e2d

The scheduler is also entered via the ENSKED or EDISMS monitor JSYS: 1e3

on a HALTF, DISMS, or other user JSYS that changes the state of the job 1e3a

And via a SCHEDR or SCHEDP monitor JSYS on a page fault 1e4

Garbage collection. 1f

GCCOR says, if the number of pages on replacble queue, plus the number of writes in progress is less than minimum (NRPMIN), or "essential GC requested", do it, otherwise forget it and return. 1f1

Collection consists of running through core status tables and writing pages on the drum, disk, or flushing them if writing is not necessary. 1f2

This is the only swapout mechinism in the system, except for XGC (see below). 1f2a

"Essential GC" is requested when there are no free pages and one is needed for a page fault. 1f2b

Pages are not written if: 1f3

They are assigned to a job in the balance set. 1f3a

pages assigned to forks designated for removal are written. 1f3a1

If a page is locked 1f3b

If a page write is in progress for that page. 1f3c

Page fault. 1g

A page fault is a trap from the pager, not an interrupt. Generally the code to handle them is hairy. Page faults are used recursively sometimes. 1g1

The most interesting page fault is for a not-in-core page: 1g2

This results in going to NIC, who will either let you have the new page or krunch you. 1g2a

The code is not straight forward, but generally is as follows: 1g2b

(NIC) 1g2b1

Update Tav 1g2b1a

(NIC3A) 1g2b1b

IF Tav > PTav GOTO NIC3 (plan to krunch his working set 1g2b1c

Adjust his reserve to squeeze in one more page if necessary 1g2b1d

If he's on a high queue or WS < TOTRC2 let him have the page 1g2b1e

Otherwise, if forks are on the go list, GOTO NIC3 1g2b1f

IF there are no other forks in the balance set, let him have the page 1g2b1g

If the balance set has room for another page, let him have it. 1g2b1h

NOTE: "let him have the page" means if WS > NPMAX, GOTO NIC3. Otherwise get the page in core, assign it to him, etc. 1g2b1i

(NIC3) 1g2b2

If he hasn't used all of his resere, adjust to current size +1 and GOTO NIC3A 1g2b2a

Call XGC to krunch his working set. 1g2b2b

XGC writes out old pages in his working set. 1g2b2b1

Re-adjust his reserve and GOTO NIC3A 1g2b2c

The page read is initiated, and the fork is put in a page wait state, but remains in the balance set. Rescheduling takes place. 1g2c

<SRI-MOD> Faults on pages that are read from the disk result in the fork being "designated for removal" from the balance set so that they don't take up memory. 1g2c1

DIA 22-JUL-71 9:31 7419

The TENEX Scheduler

(J7419) 22-JUL-71 9:31; (Expedite) Title: Author(s): Don I. Andrews/DIA; Distribution: Walter L. Bass, Mimi S. Church, William S. Duvall, Douglas C. Engelbart, J. D. Hopper, Bruce L. Parsley, William H. Paxton, Harvey G. Lehtman, Charles H. Irby, John T. Melvin, Richard W. Watson, Don C. Wallace, Kenneth E. Victor, Dirk H. van Nouhuys/WLB MSC WSD DCE JDH BLP WHP HGL CHI JTM RWW DCW KEV DVN; Sub-Collections: ARC; Clerk: BER;
Origin: <ANDREWS>SCHED.NLS;3, 21-JUL-71 11:44 DIA ;

JWM 22-JUL-71 10:04 7421

CHUCK, JOHN MELVIN WAS GOING TO GET ME THE DOCUMENTATION OF TREE/A-META. DO YOU KNOW IF IT IS READY? I D HOPED TO GET AT IT THIS WEEKEND.

1

(J7421) 22-JUL-71 10:04; Title: Author(s): John W. McConnell/JWM; Distribution: Charles H. Irby/CHI; Keywords: ; Sub-Collections: NIC; Clerk: JWM;

DVN 22-JUL-71 16:10 7438

July TNLS Course

Monday and Tuesday of this week with support from RWW I taught the class in TNLS discussed in (Journal, 7399,2) to two students, Alex McKenzie who is liaison man at BBN and is also writing the Network Resource Notebook and Dave Growthe who was substituting for liaison man James Madden from Illinois. Both came with considerable knowledge of ARC and were generally bright and sophisticated. 1

McKenzie showed special interest in Net operations and logged in through the Net to us, BBN, and UCLA. He took the opportunity to discuss various matters of Net operation with Dick Watson and Jeanne North. He spoke of writing the Resource Notebook on line in NLS. 1a

Growthe showed special interest in L-10 and TREE-META with which he was already familiar and in problems of moving NLS to other machines and/or large blocks of code to our machine. He discussed these questions with Bill Paxton. 1b

There were some rough edges to the course similar to the rough edges in the previous course. A bug appeared in TNLS links for a while and, because of changes in the Journal command language the previous week, I didn't know exactly what I was doing when I went to guide people through a Journal entry. However, we succeeded. 2

It came home to me that I or whoever teaches such courses should make a point of limbering up his fingers with TNLS a couple of hours a day the previous few days, not so much to learn things as to refresh confidence in what you know in order to handle examples readily and answer cleanly student's questions which are often very ingenious. 2a

DVN 22-JUL-71 16:10 7438

July TNLS Course

(J7438) 22-JUL-71 16:10; Title: Author(s): Dirk H. van Nouhuys/DVN; Distribution: Richard W. Watson, Barbara E. Row/RWW BER; Keywords: TNLS Training; Sub-Collections: ARC; Clerk: BER;
Origin: <VANNOUHUYS>TNLSCLASS.NLS;1, 22-JUL-71 16:09 BER ;

CATALOG PRODUCTION AUTOMATON DESIGN PROPOSAL

CATALOG PRODUCTION AUTOMATON DESIGN PROPOSAL

I. INTRODUCTION 1

In recent months a major portion of NIC personnel resources has been committed to the process of producing updated versions of the NIC Catalog. To facilitate this process and permit a more efficatious allocation of resources, the NIC has requested, as "buyer," that user-system features be provided through NLS that will allow automatic updating and production of the NIC Catalog. This document is a design proposal for a "Catalog Production Automaton" (CPA) which can be incorporated into NLS to provide the required capabilities. 1a

II. REQUIREMENTS 2

Dick's memo "Catalog Requirements" (Journal,7263) outlines the process by which the NIC catalogs and indices are currently produced and provides guidelines for the design of the CPA. My working set of design requirements is described below. 2a

The following design goals are assumed: 2b

(1) It should be possible to specify the entire catalog production procedure in a flexible and comprehendable manner, e.g., with a special-purpose programming language. 2b1

(2) In the absence of errors the CPA should be capable of carrying out the catalog production procedure without manual intervention. 2b2

(3) The CPA should be able to recover automatically from error conditions such as disc errors in output files and should be capable of being restarted manually following fatal errors. 2b3

(4) The CPA should provide adequate feedback during its operation to permit the user to monitor the state of the process and to make performance evaluations. 2b4

(5) The CPA should be capable of scheduling its own operation so as to provide minimum interference with on-line users of the system. 2b5

The CPA must interface cleanly with the following system entities: 2c

(1) NLS 2c1

-- as the user's control port to the CPA. 2c1a

-- as the CPA's normal system residence and interface to other system entities. 2c1b

-- as a manipulator for performing operations on NLS files. 2c1c

(2) Collector-Sorter-Merger (CSM) -- for updating and ordering catalog files. 2c2

(3) Analyser-Formatter (AF) -- for compiling the various programs used for document formatting. 2c3

(4) Output Processor -- for formatting files for printing. 2c4

(5) Hardcopy production facilities -- initially the Line Printer (LPT), eventually on- or off-line microform production devices. 2c5

(6) TENEX -- as system host for special file and scheduling operations. 2c6

The CPA must provide clean user interface capabilities in the following areas: 2d

(1) Specification of the procedure to be followed in producing the catalog. 2d1

(2) Coordination of CPA control specifications with OP, AF, CSM, and NLS control specifications needed for describing the catalog production procedure completely. 2d2

(3) Specification of TENEX operations needed to schedule the CPA process and to perform special file operations (such as deleting intermediate files no longer needed). 2d3

III. CONSIDERATIONS 3

Time is of the essence, and it seems that a simple system which will satisfy the immediate requirements -- with capability for subsequent expansion -- will prove more cost/effective than a more elaborate or general system that would require an expenditure of resources which might better be conserved for application in other critical areas. I propose to adopt this position so as to provide NIC with a usable system as soon as possible. 3a

One condition that might modify this stand is the fact that a new Collector/Sorter/Merger is currently being designed, and the fast CPA solution will probably involve interfacing to both the old ColSort and the new CSM. 3a1

This probably will not be a problem, as changing the interface should be relatively simple and the number of CPA users will be so small that reprogramming should not be a major expense. 3a1a

Also, it may not be a good idea to hold up the CPA while the new CSM is being designed and implemented, as the CSM job is probably the harder of the two, at least in terms of final debugging. 3a1b

Even if we do adopt this position, we should be open to discussions regarding more general future developments along the lines of the CPA -- i.e., processes involving complicated interactions among and operations on a number of seperate NLS files. 3a2

One of the most basic design considerations affecting user interaction is determining the kind of language to be used in specifying the CPA procedure to be followed. 3b

There are three prime factors involved in choosing this language: 3b1

(1) the esthetics of the language -- how elegently and compactly the procedure can be specified in the language 3b1a

(2) the clarity and ease of use of the language -- how simple it is for a user to write a CPA program to do what he wishes 3b1b

(3) the ease of implementation for the language -- how straightforward it is to specify the syntax of the language and program and debug a compiler for it. 3b1c

My own basic inclination is towards an elegent language in which complicated sequences of operations can be expressed in a single, algebra-like statement. 3b2

However, both the necessity for a less expensive implementation and the desirability of straightforward syntax (operations specified step-by-step in the language) call for a simpler language. 3b2a

Also, since the CPA will likely be used only by a few people and only for production jobs requiring infrequent changes, execssive frills should be avoided whenever possible. 3b2b

These considerations led to the proposed language design outlined below. 3b3

IV. USER INTERACTION 4

CPA Control Language 4a

Discussion 4a1

The CPA control language will be translated from NLS files with a compiler generated using META. 4a1a

The compiled programs will be stored and executed within NLS. 4a1b

Programs can be started (or restarted) at any labeled statement. 4a1c

During execution, the CPA will print out each program statement as it is executed along with the start-time and elapsed time for each operation. 4a1d

Syntax 4a2

<cpa-program> = $ ((<labeled> / <statement>) ';) FINISH; 4a2a

A CPA program is any number of labeled or unlabeled statements, with each statement terminated with a semi-colon and the entire program terminated with "FINISH". 4a2a1

<labeled> = '(<identifier> ') ': <statement> ; 4a2b

Statements may be labeled, and these labels provide entry points for executing the program starting at intermediate points -- e.g., for restarting after an error has been corrected. 4a2b1

<statement> = <compile> / <colsort> / <delete> / <output> / <print> / <append> / <comment> / <quit> ; 4a2c

There is a seperated statement type for each of the eight permitted "operations." 4a2c1

<compile> = COMPILE <link> ; 4a2d

Compile (using L10) the analyser/formatter program starting at the location addressed by the link. 4a2d1

<colsort> = COLSORT <link> ; 4a2e

Execute the Collector/Sorter/Merger operaton specified in the text starting at the statement addressed by the link. 4a2e1

This version will be used with the proposed new CSM which compiles its specs from an NLS file. 4a2e2

<colsort> = COLSORT BEGIN $ (<cs-statement> ';) END ; 4a2f

<cs-statement> = INPUT <file-list> / OUTPUT <file-name> / VIEWSPECS 1$L / DELETE KEYS / SORT / MAX <number> / LENGTH / GO ; 4a2g

These statements set the colsort parameters as if they had been set manually in NLS. 4a2g1

<delete> = DELETE <file-list> ; 4a2h

The indicated files are deleted by TENEX. 4a2h1

<output> = OUTPUT <file-list> TO <file-name> ; 4a2i

The indicated files are formatted by the Output Processor into the indicated text file. The files in the file list are treated as if they were strung together into a single file, and only the first file's origin statement is formatted. 4a2i1

<print> = PRINT <file-name> ; 4a2j

The indicated text file is copied to the line printer. 4a2j1

<append> = APPEND <link> TO <link> ; 4a2k

The text of the statement specified by the first link is appended to the statement specified by the second link. (This will allow directives to be inserted into the collected files for output processing.) 4a2k1

<comment> = COMMENT <text-string-excluding-semi-colon> ; 4a2l

The text string is printed out on the controling teletype at run-time. 4a2l1

<quit> = QUIT ; 4a2m

Execution of the CPA program is terminated, and the user is returned to NLS; the program remains loaded in NLS and can be restarted manually, if desired. A quit instruction is automatically inserted at the end of every CPA program. 4a2m1

NLS Commands 4b

Discussion 4b1

These descriptions are merely suggestive, as final choice of commad syntax must be made in consultation with NLS people so as to fit into the overall NLS command structure. 4b1a

Syntax 4b2

Execute Compile CPA Program (<statement-designator> / CA) 4b2a

The CPA program starting at the specified statement (current statement, if none is specified) is compiled, and the resulting code is loaded into the appropriate NLS data area. If there is a syntax error, NLS will post the message "Syntax Error: Type CA". 4b2a1

Goto CPA Program (<program-address> / CA) 4b2b

The CPA program is executed starting at the specified location (program start, if no address is specified). 4b2b1

V. DESIGN AND IMPLEMENTATION PROCESS 5

Introduction 5a

Due to the large number of system elements with which the CPA must be coordinated, it will be necessary to bring several people into the design and implementation process as "consultants". 5a1

I will have to greatly expand my awareness of how various parts of the NLS/TENEX system operate and interact so as to be able to design clean interfaces and minimize the danger of introducing destructive conditions. 5a1a

These same factors may result in unpredictable delays due to difficulties in achieving a successful implementation. 5a1b

Below I have tried to outline the steps I see as necessary for completing the design and implementation of the CPA. 5a2

For each step I have estimated the resources required for completing that step in terms of my own time as related to system availability and time required from other persons acting in consultant roles. (All times are expressed in hours.) 5a2a

I have expressed my expected requirements for system access in four categories relating to different types of tasks and system service capabilities: 5a2b

"Input" -- on-line time needed for entering first drafts of programs and documentation into the system. Also includes time for studying on-line materials and some time for off-line consultation. Estimated times are for prime daytime system use; subtract 10-20% for evening use and add 20-40% for off-line composition when system is down (plus a substantial amount of clerical time for final input). 5a2b1

"Edit" -- on-line time for changing programs and editing documentation. Estimated times are for a mix of prime and evening use; add or subtract 20% for any substantial shift either way. 5a2b2

"Debug1" -- on-line time for iterative program compilation, loading, testing, and debugging. Estimated times are for evening use with no other conflicting users; add 200-1000% for any substantial shift of this activity into prime time. 5a2b3

"Debug2" -- on-line time for debugging of difficult bugs with assistance of "consultants." Estimated times are for prime daytime use, subject to availability of needed consultants. Subtract 25%-75% for parts which can be done during evening time. 5a2b4

Steps 5b

(1) Obtain assigned program and data areas in NLS's address space and add CPA commands to NLS command language. 5b1

Time: Input 4 Edit 2 Debug1 4 Debug2 2 Total 12 5b1a

Consultants: WHP 2 CHI 2 5b1b

(2) Design and build a primitive compiler implementing only Comment and Quit statements to permit debugging of basic CPA-NLS interactions. 5b2

Time: Input 8 Edit 4 Debug1 12 Debug2 4 Total 28 5b2a

Consultants: WHP 4 DIA 2 HGL 1 5b2b

(3) Implement and debug APPEND statement (makes use of file links). 5b3

Time: Input 12 Edit 2 Debug1 12 Debug2 2 Total 28 5b3a

Consultants: WHP 2 DIA 2 CHI 1 5b3b

(4) Implement and test COMPILE Statement. 5b4

Time: Input 4 Edit 0 Debug1 4 Debug2 1 Total 9 5b4a

Consultants: CHI 1 5b4b

(5) Design and build interface to output Processor and implement OUTPUT statement (without file lists at first). 5b5

Time: Input 8 Edit 2 Debug1 8 Debug2 2 Total 20 5b5a

Consultants: CHI 2 5b5b

(6) Add file list capability and incorporate into OUTPUT statement. 5b6

Time: Input 4 Edit 2 Debug1 8 Debug2 1 Total 15 5b6a

Consultants: WHP 1 5b6b

(7) Build interface to Collector/Sorter, add COLSORT statement to CPA control language, and debug. (Interface would be to new Collector/Sorter/Merger, if it is ready by this time, otherwise to old Collector/Sorter.) 5b7

Time: Input 16 Edit 4 Debug1 12 Debug2 4 Total 36 5b7a

Consultants: WHP 1 WSD 4 JDH 2 5b7b

(8) Test CPA implemented so far with a representative catalog production job, and correct any problems. 5b8

Time: Input 2 Edit 0 Debug1 4 Debug2 0 Total 6 5b8a

Consultants: RWW 2 JBN 2 5b8b

(9) Interface CPA to Exec and implement DELETE and PRINT statements. 5b9

Time: Input 4 Edit 0 Debug1 4 Debug2 1 Total 9 5b9a

Consultants: WSD 1 JTM 1 5b9b

(10) Prepare final documentation and submit CPA package to NIC for acceptance testing, correcting any bugs uncovered. 5b10

Time: Input 4 Edit 2 Debug1 8 Debug2 0 Total 14 5b10a

Consultants: RWW 2 JBN 2 5b10b

(11) If necessary, prepare interface to new Colletor/Sorter/Merger and debug. 5b11

Time: Input 4 Edit 0 Debug1 8 Debug2 2 Total 14 5b11a

Consultants: JDH 4 5b11b

(12) TOTALS (including interfacing to both old CS and new CSM) 5b12

Time: Input 70 Edit 18 Debug1 84 Debug2 19 Total 191 5b12a

Consultants: WHP 10 CHI 6 DIA 4 WSD 5 JDH 6 RWW 4 JBN 4 HGL 1 JTM 1 (Total 41) 5b12b

<JOURNAL>7451.NLS;1, 23-JUL-71 15:21 WLB ; (Expedite) Title:
Author(s): Walter L. Bass/WLB; Distribution: Richard W. Watson, Jeanne
B. North, James C. Norton, Douglas C. Engelbart, Walter L. Bass, J. D.
Hopper, Charles H. Irby, William H. Paxton/RWW JBN JCN DCE WLB JDH CHI
WHP; Keywords: ; Sub-Collections: ARC; Clerk: WLB;
Origin: <BASS>OPA.NLS;11, 23-JUL-71 14:58 WLB ;

CATALOG PRODUCTION AUTOMATON DESIGN CHANGE PROPOSAL

CATALOG PRODUCTION AUTOMATON DESIGN CHANGE PROPOSAL

INTRODUCTION 1

Dave Hopper and I met Wednesday afternoon (7/28/71) to coordinate the interface between design of the CPA and design of the new Collect/Sort/Merge package for NLS. In this memo, I will try to outline the major points of our conversation and put forth some reccommendations for modifying the CPA design presented in <Journal,7451,1:hx>. 1a

COMPILER CONSIDERATIONS 2

Dave and I have independently reached the conclusion that the control languages for both the CPA and the CSM are so simple that using a META-produced compiler is needlessly expensive in terms of implementation effort and execution overhead. 2a

The linguistic analysis features of the compilers would be used principally for decoding commands and parsing file names (links); however, command decoding could be done just as well using the string analysis capabilities of L10, and the machinery for parsing links is already present in NLS. Consequently, using compilers would buy us little, if anything, while introducing several additional complications into the design and implementation process. 2a1

The only advantage of using compilers that we could think of was that by passing a CPA or CSM "program" through a compiler, a user would know whether the syntax of his program were correct before starting execution (during which he would undoubtedly be absent from the console so as to prevent interactive debugging). 2b

If this syntax-checking feature seems to be sufficiently desireable, it will be possible to provide it in a non-compiler implementation at a relatively low cost. This could be done by designing the CPA and CSM so that they can be "disarmed" -- i.e., run with all of the processing routines no-oped. 2b1

PROPOSAL TO MERGE CPA AND CSM INTO A SINGLE PACKAGE 3

Dave and I have tentatively come to the conclusion that the CPA and CSM should be merged into a single operational subsystem of NLS. There are several considerations influencing this decision; these are discussed below. 3a

The basic reason for implementing the CPA is to provide a capability for specifying NLS operations involving several

steps, each of which can cause a complicated operation to be carried out on one or more files. 3a1

To provide this service, the CPA needs a control language which gives the user two basic capabilities: 3a1a

(1) The ability to specify a sequence of operations selected from a limited repertory. 3a1a1

(2) The ability to talk about sets of files, some of which exist at CPA start-up time and some of which are generated during execution, in a convenient and (hopefully) concise way. 3a1a2

It turns out that the CSM must provide precisely these same linguistic capabilities to permit specification of a complete CSM operation, since such an operation may itself consist of several steps, each of which involves one or more sets of files. 3a1b

In this light, it seems wasteful to design two seperate subsystems with such similar capabilities -- if there is a good way to combine them into a single subsystem without incurring any unnecessary "losses." 3a1b1

The most complicated interaction which the CPA must indulge in is with the CSM, and this interaction apparently must be two-way to guarantee proper file-referencing. 3a2

Since the CSM produces an unpredictable number of files during its operation, it must communicate the compelete name and version number back to the CPA to enable subsequent operations. Since there is no other subsystem which must communicate such complicated information back to the CPA, it would seem reasonable to put both the CPA and the CSM features into a single subsystem "package." 3a2a

Architecturally, it seems a little like a case of "the tail wagging the dog" to create a CPA subsystem whose major purpose is to drive the CSM rather than just adding some additional capabilities to the CSM itself. 3a3

This is principally a matter of personal aesthetics, and I feel very uneasy about the previous CPA design in light of the way that the CSM design is shaping up (which looks very good so far). 3a3a

If all the principals agree with this design change, I will begin to work with Dave on designing the new CSM control language and integrating the needed auxiliary functions into the CSM package. 3b

<JOURNAL>7465.NLS;1, 4-AUG-71 11:14 WLB ; (Expedite) Title:
Author(s): Walter L. Bass/WLB; Distribution: Walter L. Bass, James C.
Norton, Jeanne B. North, Richard W. Watson, Charles H. Irby, William H.
Paxton, J. D. Hopper, Douglas C. Engelbart/WLB JCN JBN RWW CHI WHP JDH
DCE; Keywords: ; Sub-Collections: ARC; Clerk: WLB;
Origin: <BASS>MEMO.NLS;3, 4-AUG-71 11:08 WLB ;

Section 1. The TENEX OPERATING SYSTEM AND EXECUTIVE

1

INTRODUCTION 2

TENEX is an interactive timesharing system produced by Bolt Beranek & Newman for the DEC PDP-10. NLS users must use certain facilities of the TENEX system through the system's EXECUTIVE language. The TENEX facilities of primary interest to the NLS user are access to the NLS subsystem itself and the file system. 2a

When the terminal is connected to the ARC PDP-10 computer via the Network, the TENEX system will print the message: 2b

ARC-TENEX xxxxx date EXEC xxx 2b1

where: xxxxx = information which identifies the current version of the system 2b2

(If the user is not connected through the Network, he must type control C (↑C) to access TENEX.) 2c

TENEX responds that it is ready to accept information by typing the character '@'. Before the user can perform any tasks on the system he must first identify himself using the LOGIN command: 2d

log CR *[CR]*
[(user)] USERNAME CR *[CR]*
[(password)] PASSWORD CR *[CR]*
[(account #)] ACCOUNT NO. CR *[CR]* 2d1

A quicker version of the Login sequence may be achieved by typing Space (SP) instead of CR after each entry except the last (account number). 2e

log SP USERNAME SP PASSWORD SP ACCOUNT NO. CR 2e1

where: 2f

USERNAME = 1-39 alphanumeric characters (excluding the characters ; and .)
PASSWORD = 1-39 alphanumeric characters (excluding the characters ; and .) that are not echoed by the system
ACCOUNT NO.= 1-39 characters; (#1 is currently used for all users on the NLS system) 2f1

When the user has successfully logged in to the system the following message is printed: 2g

JOBxx ON TTYyy date time 2g1

where: 2h

xx = job number assigned to terminal during terminal session 2h1

yy = terminal identification number 2h2

If a user fails to successfully login within two minutes, the system automatically prints the message 2i

AUTOLOGOUT
KILLED JOBxx, TTYyy, AT date time
USER time1 IN time2 2i1

where: 2j

time1 = total computer time used
time2 = total terminal time used 2j1

EXECUTIVE LEVEL COMMAND CHARACTERS 3

The following special characters are recognized by the TENEX system as commands and are used in conjunction with TENEX command words (These do not function as commands in NLS): 3a

ALT MODE/ESCAPE (ALT/ESC) 3a1

This key forces recognition of any user entry as far as possible within an input item. For example, if the user types "LOG ALT" the system will print "IN". If insufficient characters are entered for system recognition, the system will ring the bell or print the character "?" (depending on the device); the user should respond with additional characters. 3a1a

CONTROL F (↑f) 3a2

This key forces recognition of the individual parts (fields) of a filename entry. For example, the user may enter the name of the file "DOE.NLS;1" by typing "D ↑f N ↑f 1" assuming that it is the only NLS file in the current directory whose name field begins with a "D". The user may move from field to field within filename by pressing the ↑f key repeatedly. 3a2a

SPACE (SP) 3a3

This key forces recognition in the same manner as the ALT or ESC keys but does not cause the remainder of the entry to be echoed at the terminal. For example, if the user types "LOG SP", the system prints out only "LOG ", but interprets it as "LOGIN". 3a3a

CR/RETURN (CR) 3a4

This key forces recognition as do the ALT and SP characters and also confirms the current command for execution. Most TENEX commands must be terminated with CR in order to be executed. 3a4a

ASTERISK (*) 3a5

The asterisk character may be used in any of the fields of a filename entry to designate all possible entries in that field. For example, a filename "mmm.*;*" indicates all files whose name is mmm. The * may be used on any and/or all fields of filename. 3a5a

EXECUTIVE COMMAND SET 4

The following commands are a subset of all the commands available in the EXECUTIVE language. Only those basic commands necessary to the NLS user are covered here. 4a

Although the entire command word is shown for each command, the first few characters are usually sufficient for recognition by the system. 4b

DIRECTORY COMMAND AND SUBCOMMANDS 4b1

The directory command causes the system to print the names of all the files in the user's file directory. 4b2

directory CR 4b2a

The user may also view the contents of another user's directory by using another form of the DIRECTORY command: 4b3

directory SP <OTHER DIRECTORY'S NAME> CR 4b3a

The user may access subcommands of the DIRECTORY command using the form of the DIRECTORY command: 4b4

directory SP , CR 4b4a

When the user has accessed the DIRECTORY subcommand level the system prints the characters "@@". The following commands may then be issued by the user: Note that the field EMPTY in each of the following subcommands indicates that the user may respond to "@@" immediately with a Carriage Return. 4b4b

SIZE
The Size subcommand causes the system to print the total number of pages of each file in the user's directory. One page is approximately equal to one line printer or typed page. 4b4b1

@@size CR *[CR]*
@@ EMPTY CR 4b4b1a

EVERYTHING
The Everything subcommand causes the system to print all system-maintained information about a file. 4b4b2

@@everything CR *[CR]*
@@EMPTY CR 4b4b2a

DELETED (FILES ONLY)*]*
The Delete subcommand causes the system to print a directory of all the user's deleted files at the terminal. 4b4b3

@@deleted *[files only]* CR *[CR]*
@@EMPTY CR
4b4b3a

The user may specify combinations of multiple subcommands by entering a subcommand instead of the empty field when the system echoes a CR. For example, the user may obtain a list of all information (EVERYTHING) about deleted files only by using the following sequence of subcommands: 4b5

@@ eve CR
@@ del CR
@@ CR 4b5a

The user may leave the DIRECTORY subcommand mode by following any subcommand with two successive CR's.
4b6

CONNECT COMMAND 4b7

The connect command enables the user to access files in another user's directory without having to preface any files in that directory by the directory's name in anglebrackets. 4b8

connect *[(to directory)]* SP DIRECTORY CR 4b8a

Where DIRECTORY = the name of the directory to which the current user will be connected. 4b9

After this command is executed the user may access any of the files in the directory to which he is connected as though they belong to the directory under which he logged into the system. However, as he is connected to another directory, he cannot access files in his own directory without prefacing those files by his own directory name enclosed in parentheses. 4b10

Although a user is connected to another directory, when he edits a file in NLS, a partial copy is created under his own directory name just as though he were accessing another user's file by conventional means (i.e. prefacing the file name by the directory name enclosed in anglebrackets). 4b11

DELETE COMMAND 4b12

The DELETE command removes a file from the user's directory. The file is not destroyed but cannot be copied, does not appear in the directory list, and cannot be loaded into the NLS subsystem. 4b13

delete SP FILENAME CR 4b13a

The user may enter filename using any of the characters that force system recognition of a file name as described earlier in this section. 4b14

A list of all deleted files is maintained by the system. Use the DIRECTORY subcommand DELETED(FILES ONLY) to obtain this list. 4b15

Files may be permanently removed from the system by the user with the EXPUNGE command. All deleted files are also expunged by the system when incremental dumps are taken (daily) at the Augmentation Research Center; deleted files belonging to a particular directory are expunged when a user issues the LOGOUT command when logged in under that directory name. 4b16

EXPUNGE COMMAND 4b17

The EXPUNGE command erases deleted files from file storage. When this command is executed all deleted files belonging to the user are affected. 4b18

expunge CR 4b18a

To obtain a list of deleted files see the DIRECTORY DELETED(FILES ONLY) subcommand. 4b19

To selectively expunge deleted files, the user must first undelete (see the UNDELETE command) any files that are not to be destroyed before issuing the EXPUNGE command. 4b20

All deleted files are also expunged by the system when incremental dumps are taken (daily) at the Augmentation Research Center; deleted files belonging to a particular directory are expunged when a user issues the LOGOUT command when logged in under that directory name. 4b21

UNDELETE COMMAND 4b22

The UNDELETE command restores a deleted file to the user's file directory. 4b23

undelete SP FILENAME CR 4b23a

filename may be entered using the characters that force system recognition of a file name as described earlier in this section. 4b24

To obtain a list of deleted files, use the DIRECTORY DELETED(FILES ONLY) subcommand. 4b25

This command may be used before EXPUNGE to selectively expunge files from the system. 4b26

RENAME COMMAND 4b27

The user may change the name of a file by using the RENAME command. 4b28

rename *[*(existing file)*]* FILENAME *[*(TO BE)*]* FILENAME CR 4b29

Where first FILENAME = the file whose name will be changed to second filename 4b30

second FILENAME = the destination filename 4b31

After the user types CR after the destination filename, the system responds with one of the following messages: 4b32

*[*NEW FILE*]*
This message is issued if the destination file does not currently exist in the user's file directory. In response the user may type CR to confirm the creation of a new file, or type the name of another file. 4b32a

/NEW VERSION/
This message is issued if the destination file already exists in the user's file directory but with a different (lower) version number. (If the user had used one of the characters that force recognition when entering the filename, TENEX would have generated a version number that is one greater than the highest version number current for the filename.) In response the user may type CR to confirm the new version number or type a previous version number causing the earlier version to be replaced by the source file. 4b32b

/OLD VERSION/
This message is issued if the destination file already exists in the user's file directory with the same version number. In response the user may type CR to confirm writing over the old file, or enter a new version number or a new filename. 4b32c

SHUT COMMAND 4b33

The shut command closes all open files. 4b34

shut CR 4b34a

FULLDUPLEX/HALFDUPLEX COMMANDS 4b35

These commands control how the computer at ARC sees local terminals over the Network. 4b36

fullduplex CR	characters entered by the user are transmitted over the Network and then echoed by NIC	4b36a
halfduplex CR	characters entered by the user are not echoed by NIC but are echoed by the TELNET program at the local site.	4b36b

The default value for Network users is local echoing - halfduplex. 4b37

LINK COMMAND 4b38

The link command enables the user to communicate with another user who is currently connected to the system. 4b39

link *[(to)]* USERNAME CR
TERMINAL NO. 4b39a

The user may specify either the name of another user or the terminal at which the other user is running. 4b40

When this command is executed, the user may communicate the the user specified by typing the character ";" followed by any series of characters (message) and terminated by CR. As the user types the message, it will appear at the other user's terminal but the appearance of this output will have no effect on the other user's job. The other user may respond in turn by using the semicolon followed by a message and CR. Simultaneous messsage sending by both parties causes characters to be interleaved. 4b41

A link may be "broken" by using the Break Links command described below. 4b42

BREAK LINKS COMMAND 4b43

This command breaks any links that the user nas established with other users. 4b44

```
   break /(links)/ CR
```
4b44a

SYSTAT COMMAND 4b45

The systat command is a query to the system requesting information about current system usage. 4b46

```
   systat CR
```
4b46a

When this command is executed, the system responds with the following information: 4b47

```
   UP  nours:minutes:seconds
```
4b47a

```
   JOB  TTY    USER        SUBSYS
   ...  ...    ....        ......
```
4b47b

For example: 4b48

UP 10:49:12 4b49

```
JOB  TTY    USER        SUBSYS
 1   12     TOMAS        TNLS
 2    4     SMITH        EXEC
 3    9     JONES        NLS
```
4b50

OTHER STATUS COMMANDS 4b51

The following commands enable the user to obtain selected information about the status of the system and his job: 4b52

command	information printed by the system	
-------	---------------------------------	4b52a
dskstat CR	disk status - number of pages assigned to current user	4b52b
filstat CR	file status - files currently open and directory to which user is currently connected	4b52c
jobstat CR	job status - job number, user name, device assignment	4b52d
runstat CR	run status - current user job activity	4b52e
usestat CR	user status - total CPU time used in total terminal time	4b52f
version CR	current version of TENEX	4b52g

WHERE COMMAND 4b53

This command enables the user to determine the terminal and job number of any user currently on the system. 4b54

where *[(is user)]* USERNAME CR 4b54a

When this command is executed, the system print the job number and terminal number of the USERNAME specified. 4b55

DAYTIME COMMAND 4b56

The Daytime command causes the system to print the current date and time. 4b57

daytime CR 4b57a

ENTERING NLS 4b58

In order for the user to enter NLS, he must use the EXECUTIVE command NLS. 4b59

```
@nls CR
ID: USER IDENTIFICATION CA
device:  N[et-tty]
```
4b59a

NOTE: if the user's local TELNET program transmits only upper case characters, the user should respond to the device request with "33". 4b60

A Network user may establish his user identification by contacting the NIC technical liaison at his site. 4b61

An asterisk (*) in the margin is NLS's signal that it is awaiting a command. The asterisk is printed whenever NLS completes a command -- in other words, if the asterisk is not printed, NLS is not yet ready to process another command. 4b62

CONTINUE COMMAND 4b63

If the user has used the control C (↑C) character to leave the NLS subsystem, the CONTINUE command enables the user to reenter the NLS subsystem. 4b64

```
continue CR
```
4b64a

When the user is returned to the NLS subsystem, the status and contents of the NLS area will not have been changed. 4b65

When the user is returned to the NLS subsystem, NLS will not respond with its prompt character "*". However, it will accept commands as long as there was no output operation in progress when the user left the subsystem. 4b66

REENTER COMMAND 4b67

If the user has used the NLS command, Execute Quit, to leave the NLS subsystem, the REENTER command enables the user to resume work in NLS. 4b68

reenter CR 4b68a

When the user is returned to NLS, the status and contents of the NLS area will not have been changed. 4b69

RESET COMMAND 4b70

The reset command closes any open files and resets NLS. It appears as though the user has just logged in to the system. 4b71

reset CR 4b71a

After this command is executed, the user may not enter NLS without using the NLS command. 4b72

LEAVING THE TENEX SYSTEM - LOGOUT COMMAND 5

The LOGOUT command enables the user to leave the system and causes certain accounting information to be printed at the terminal. 5a

logout CR 5a1

When this command is executed all deleted files belonging to the directory under which the user is logged in are expunged and the system prints the message: 5b

KILLED JOBxx,USER username, ACCT account no.,TTY yy, AT date time
USED time1 IN time2 5c

where: 5d

time1 = total computer time used
time2 = total terminal time used 5d1

Section 2. FILE STRUCTURE, CONTENT, AND INPUT/OUTPUT OPERATIONS

1

Part 1. FILE STRUCTURE

2

When working in NLS, one is at all times constructing, studying, or modifying a file. NLS files have a hierarchical, tree, or outline structure.

2a

2b

```
0 ...                                                    2b1
1 ...
   1a ...
   1b ...
      1b1 ...
      1b2 ...
      1b3 ...                                            2b2
2 ...                                                    2b3
3 ...
   3a ...
   3b ...
   3c ...
      3c1 ...
   3d ...
      3d1 ...
      3d2 ...
         3d2a ...
         3d2b ...
         3d2c ...                                        2b4
4 ...
   4a ...
   4b ...                                                2b5
5 ...
   5a ...
      5a1 ...
      5a2 ...
         5a2a ...
   5b ...                                                2b6
```

2c

FIGURE 1. Hierarchical File Structure

2d

It would be difficult to overstate the importance of this structure in the design of NLS; it is correspondingly important for the user to understand the structure and its terminology. 2e

In the remainder of this discussion of file structure, note that every statement is headed by a string of digits and letters. These strings are the statement numbers associated with the file structure; they have been suppressed from the rest of the document, but are printed here as an example. Also, the reader is invited to observe the way this document is formatted; the indentation of statements reflects "level" in the structure. 2f

3 OVERALL FILE STRUCTURE 3

3a Every NLS file is made up of STATEMENTS, entities which may contain any sort of text (every paragraph and heading in this document is a statement). 3a

3a1 Every NLS file has an ORIGIN STATEMENT or "zero statement". (The origin statement has been omitted from the printout of this document). The origin statement is a "0th-level" statement (the only one in the file). 3a1

3a2 The statements immediately below the origin statement in the outline are "1st-level" statements (all section titles in this document are the 1st-level statements). 3a2

3a3 The statements immediately below the 1st-level statements are 2nd-level statements, and so forth to arbitrary depth. 3a3

4 STATEMENT NUMBERS 4

4a Every statement has a unique "statement number." This is a string of alternating fields of numbers and letters. The statement number is a primary means of addressing parts of the file in NLS commands. 4a

4a1 The first field always contains a number. 4a1

4a2 The number of fields is equal to the level of the statement. Properly speaking, the origin statement should have no statement number, since its level is 0; for convenience, however, the statement number "0" is assigned to it. 4a2

4a3 The statement number (and its following space) is NOT part of the text of the statement; it is associated with the position of the statement in the file and is subject to change when the file structure is modified by adding, deleting, or moving statements. 4a3

4b When necessary, the @ character is used in the letter fields of statement numbers as an "alphabetical zero." Thus the 26 letters and the @ can be used to form a sequence: a, b, c, ... x, y, z, a@, aa, ab, ac, ... az, b@, ba, bb, ... 4b

5 PRIMARY RELATIONSHIPS BETWEEN STATEMENTS 5

5a The following relationships between statements are defined: SUBSTATEMENT, SOURCE, SUCCESSOR, AND PREDECESSOR. These are best defined by examples, with reference to Figure 1 on page 16. 5a

5a1 SUBSTATEMENT and SOURCE refer to the relationships between statements at different levels. 5a1

5a1a Statements 1, 2, and 3 are substatements of the origin statement. Statement 1a is a substatement of Statement 1. Statements 1b1, 1b2, and 1b3 are substatements of Statement 1b. 5a1a

5a1a1 Any statement may have any number of substatements. 5a1a1

5a1a2 All first level statements are substatements of the origin statement. 5a1a2

5a1a3 Given the number of a statement, the number of a substatement is obtained by adding a field to the end of the last number. 5a1a3

5a1b SOURCE is the inverse of substatement. Statement 1b is the source of Statements 1b1, 1b2, and 1b3. Statement 3c is the source of Statement 3c1. 5a1b

5a1b1 Every statement has just one source (except the origin statement, which has no source). 5a1b1

5a1b2 Given the number of a statement, the number of the source is obtained by removing a field from the end of the first number. 5a1b2

5a2 SUCCESSOR and PREDECESSOR refer to the relationships between statements of the same level. 5a2

5a2a Statement 2 is the SUCCESSOR of Statement 1. Statement 3d2 is the successor of Statement 3d1. 5a2a

5a2a1 Not every statement has a successor. The origin statement has no successor. No statement has more than one successor. A statement and its successor always have the same level and the same source. A successor specification with a statement having no succeeding statement of the same level and source refers to the statement itself. 5a2a1

5a2a2 Given the number of a statement, the number of the successor is obtained by incrementing the last field of the first number. 5a2a2

5a2b PREDECESSOR is the inverse of successor. Statement 1a is the predecessor of Statement 1b. 5a2b

5a2b1 Not every statement has a predecessor. The origin statement has no predecessor. No statement has more than one predecessor. A statement and its predecessor always have the same level and the same source. A predecessor specification with a statement having no preceding statement of the same level and source refers to the statement itself. 5a2b1

5a2b2 Given the number of a statement, the number of the predecessor is obtained by decrementing the last field of the first number. 5a2b2

6 STRUCTURAL ENTITIES MADE UP OF STATEMENTS 6

6a Given these primary relationships -- source, substatement, predecessor, and successor -- we can define the following STRUCTURAL ENTITIES: STATEMENT, BRANCH, PLEX, and GROUP. 6a

6a1 STATEMENT has already been explained. 6a1

6a2 A BRANCH consists of a specified statement, plus all its substatements, all their substatements, etc. In the illustration, Branch 1 consists of Statements 1, 1a, 1b, 1b1, 1b2, and 1b3. Branch 1a consists of Statement 1a alone. Branch 4 consists of Statements 4, 4a, and 4b. 6a2

6a2a Branch 0, in any file, contains the entire file. 6a2a

6a3 A PLEX is made up of a specified branch, plus all the other branches that have the same source. Plex 1a and Plex 1b are the same; each consists of Branches 1a and 1b. Plex 3a consists of Branches 3a, 3b, 3c, and 3d; Plex 3b and 3c, and 3d are the same as Plex 3a. 6a3

6a4 A GROUP is a contiguous subset of a plex. It is identified by two branches, which must be in the same plex, and consists of those two branches plus all branches lying "between" them in the same plex. Group 3d2c, 3d2c consists of Branches 3d2a, 3d2b, and 3d2c. 6a4

7 SECONDARY RELATIONSHIPS BETWEEN STATEMENTS 7

7a We can now define the following relationships: HEAD, TAIL, END, UP, DOWN, NEXT, and BACK. 7a

7a1 The HEAD of a specified statement is the first statement at the same level that has the same source. The head of Statement 3d2c is Statement 3d2a. The head of Statement 5a2 is Statement 5a1. The head of Statement 3a is Statement 3a itself. 7a1

7a1a Head pertains only to members of the same plex. 7a1a

7a2 The TAIL of a specified statement is the last statement at the same level that has the same source. The tail of Statement 3d2b is Statement 3d2c. The tail of Statement 4a is Statement 4b. The tail of Statement 3c1 is Statement 3c1 itself. 7a2

7a2a Tail pertains only to members of the same plex. 7a2a

7a3 The END of a specified statement is the "last" statement in the branch defined by the specified statement. The end of Statement 3 is Statement 3d2c. The end of Statement 3c is Statement 3c1. 7a3

7a4 UP refers to the statement that is one level higher than the current statement and precedes the current statement. For example, statement 3 is up from statement 3c. 7a4

7a5 DOWN refers to the statement following the current statement that is one level lower. For example, statement 4a is down from statement 4. 7a5

7a5a Any down specification with a statement having no following statement at a lower level refers to the statement itself. Thus, excess d specifications are ignored. 7a5a

7a6 NEXT refers to the statement immediately following the current statment regardless of level or of source. For example, statement 4b is next to statement 4a; statement 5 is next to statement 4b. 7a6

7a7 BACK refers to the statement immediately preceding the current statement regardless of level and source. For example, 4b is back from statement 5. 7a7

Part 2. FILE CONTENT

8

FILE NAMES

9

The names of files in TENEX/NLS are of the following form: 9a

<DIRECTORY>FILENAME.EXTENSION;VERSION # 9a1

where DIRECTORY = 1-39 alphanumeric characters, excluding control characters, non-printing characters, period (.), and semicolon (;). This element is a TENEX user name and is required only when a user references a file belonging to a directory other than his own. 9b

FILENAME = 1-39 alphanumeric characters, excluding control characters, non-printing characters, period (.), and semicolon (;) 9c

EXTENSION = 1-39 alphanumeric characters, excluding characters control, non-printing characters, period (.), and semicolon (;) 9d

VERSION # = a numeric value (1 to 131071) 9e

The length of the entire filename (including the delimiters . and ;) must not exceed 39 characters. Otherwise, there are no restrictions on the length of any field within the total filename. 9f

TYPES OF FILES

10

There is a variety of types of files that are generated within NLS. When a user enters NLS for the first time, he is automatically assigned a file by NLS. The file is empty except for a dummy origin statement (statement 0) which contains his identification string as a filename, an extension name "NLS" and version number 1; this file is referred to an the user's "intial file". Within NLS itself, files areecreated by using the Output File and Output Device commands, see File commands described in the latter part of this section. 10a

At this point it is necessary to identify the types of files used by the NLS user. Although the user may use any identifier as an extension name, the convention generally followed by the NLS user group is to identify the type of the file by the extension name where: 10b

NLS = an NLS file 10b1

PC = a partial copy file created by NLS when the file is edited in any way 10b2

TXT = a sequential file for hardcopy output 10b3

One of these extension names is automatically supplied by the system whenever the user fails to specify extension name in a command, depending on the operation being performed. 10c

NLS FILES 10d

An NLS file is a file which may be edited or viewed in NLS. NLS files are created within NLS in two ways: when the user enters NLS for the first time, a file bearing the users identification string as its filename is created by the system; and when the user issues the Output File command and specifies a new file. 10e

PARTIAL COPY FILES 10f

Whenever an NLS file is modified a partial copy file is automatically created by the system for that file. Partial copy files have an extension name "PC" and may be used only in conjunction with an NLS file. That is, the user may not load, copy, etc. a partial copy file. 10g

When a user attempts to modify an NLS file, he is actually working on the partial copy associated with that file. Modifications are actually made to an NLS file only by operations which merge to it the contents of its partial copy. 10h

When a partial copy exists for a particular file, the file is considered "locked", i.e. no other partial copy may be made for the file. This feature prevents other users from modifying the file. A file remains locked until the user updates, outputs, or unlocks the file via the commands described in Part 3 of this section. 10i

SEQUENTIAL ACCESS FILES 10j

The hardcopy devices used by the system require sequential files, i.e., files that are processed as a sequence of characters. Any file that is to be output at a terminal requires processing by the Output Device command which essentially takes a NLS file and copies it into a sequential file for processing on a specific device. If the user, when issuing the Output Device command allows the system to 'create' an extension name for the sequential file, the extension name will be "TXT" for text (sequential) file. 10k

SYSTEM CREATION OF FILES 11

The TENEX system automatically creates files for the user under a variety of circumstances. 11a

NEW FILENAME 11a1

When the user enters the NLS system for the first time NLS automatically creates a file for him with the name "user's identification string.NLS;1". 11a2

When the user makes changes to a file in the NLS subsystem, the system automatically creates a partial copy file for the opened file. This file contains the changes made to the original file. With the NLS command Update File, the user can cause the system to add the changes back into the original and delete the partial copy. The system lists partial copies in the user's file directory as separate files with a new file name that it creates in the form (USERNAME)FILENAME.PC;#. 11a3

NEW EXTENSION NAMES 11a4

If when the user issues the Output File command in NLS, he enters a unique (to his directory) FILENAME followed by a CA. The system will automatically assign the file the extension name "NLS". Similarly, when the user issues the Output Device command, the system automatically assigns the file the extension name "TXT". 11a5

NEW VERSION NUMBERS 11a6

If, when the user outputs a file from NLS, he enters a FILENAME that exists in his directory, the system will automatically assign the file the next higher version number.

11a7

USER CREATION OF FILES 12

The user may create a new NLS file by using the Update or Output command; text files are creted by using the Output Device command. These commands are described in the next part of this section. 12a

INFORMATION IN THE ORIGIN STATEMENT OF A FILE 13

The origin statement of a named file begins with the filename, the date and time of the last modification to the file (or date of creation if it is unmodified), and the identification string of the user who modified or created it (ending with a semicolon). As explained below, this information is automatically maintained by the system. 13a

Example:
<SMITH>FILE.NLS;22, 24-MAY-71 11:50 SSS ;7, 19-14:48 SSS 13a1

Part 3. FILE INPUT/OUTPUT COMMANDS 14

LOAD FILE 15

The load file command causes the file specified to be opened and made available to the user for work in the NLS subsystem. 15a

l[oad] f[ile] FILENAME CA 15a1

Where FILENAME = the name of the file to be opened. If the user enters only the name field of FILENAME, extension NLS and the highest version number, are the default values for the remaining fields. If the file belongs to another user's directory, FILENAME must include the directory name enclosed in anglebrackets. 15a2

When this command is executed, any file and any associated partial copy currently open is automatically closed before the the file specified in the load file command is opened. 15b

If the file being loaded has an associated partial copy, the partial copy is also opened. 15b1

The user may open a file from another user's directory by prefacing FILENAME with <other user's name>. However, if the file has an associated partial copy created by the other user, the file will be "locked" to further changes by anyone but the other user (the file may be read only). In this case, the user may either request the other user to unlock the file, or he may copy the file (in EXEC) so that he has a copy in his own directory. However, when the file is copied in EXEC, the partial copy that causes the file to be locked is not also copied. 15b2

The file being opened must be an NLS file. 15b3

The user may also access files by using links, see Section. 3, Indirect Addressing. 15b4

Example: 15c

1 f myfile CA	causes the system to open the most recent version of the file myfile.nls in the current user's directory.	15c1
1 f <smith>rate.nls;3	causes the system to open a file named "rate.nls;3" belonging to the directory SMITH.	15c2

UPDATE FILE 16

The update file command causes the system to merge the contents of the current NLS file with its current partial copy. The file created by this merge can either be written onto a new version of the same file, or written over the old version of the file. 16a

u/pdate/ CA
o/(to old version)/ CA 16a1

When issuing this command the user has the option of assigning a new version number (by default), or reusing the old version number (by typing "o" (old version) before the terminating CA. 16b

Note: in general, updating to a new version is "safer" than updating to an old version. In the event of a system crash during an update to an old version, that version may be "lost" (along with its partial copy). If a crash should occur during an update to a new version, the original version and partial copy are not affected even though the new version may be lost. 16c

When updating to an old or new version, the current partial copy is automatically deleted (but not expunged) by the system. 16d

Instead of incorporating the partial copy into the current file, the user may delete all changes made to the file since the last update or output operation by using the Execute Unlock command which deletes the current partial copy. 16e

Example: If the current file is APPLE;NLS.4 16f

u o(to old version) CA	causes the current file to remain APPLE.NLS;4	16f1
u CA	causes the current file to be changed to APPLE.NLS;5	16f2

OUTPUT FILE 17

The Output File command causes the system to copy the content of the currently open file and its associated partial copy to the filename specified. 17a

o/utput/ f/ile/ FILENAME CA 17a1

Where FILENAME = the name of the file to be created. If only the name field of FILENAME is supplied, the system creates a file having the extension name "NLS" and assigns it the next highest version number. 17b

The origin statement of the destination file will contain FILENAME, the current date and time, and the identification string of the user who is creating the file. 17c

The contents of the currently open file and its partial copy are then copied into the named file. Finally, the named file is opened and the currently open file is closed and its partial copy is automatically deleted (but not expunged) by the system. Thus the Output File command always leaves you with the named file open. 17d

The difference between output File and Update File is that the file being created by Output File is ordered internally to provide more efficient access and storage. 17d1

An attempt to perform an output operation using the same filename and version number as the current file will cause the system to issue the message: 17d2

FILE BUSY 17d2a

and the command will not be executed. 17d3

When this command is executed, any partial copy associated with the file being output is deleted (but not expunged). 17d4

Example: if there is a file APPLE.NLS;4 17d5

o f apple CA creates a file APPLE.NLS;5 17d5a

EXECUTE UNLOCK 18

The Execute Unlock command deletes the contents of the partial copy associated with the current file. In effect the file is restored to its status immediately following the last update or output operation on the file. 18a

e[xecute] u[nlock] CA [filename really ?] CA 18a1

Where filename = the name of the current file 18b

An extra CA is required to terminate this command to decrease the chance of executing this command by mistake. 18c

This command is also used when the message 'BAD FILE' appears for a file. Using the Execute Unlock command enables the user to determine whether the partial copy is bad (which causes the file to appear bad) or the file itself is indeed bad. For more information about recovery from this error condition see Apppendix D. ERROR MESSAGES. 18d

OUTPUT DEVICE TELEYPE 19

The output device command causes the system to convert the current file from its random file format to a sequential format and to process it so that it may be listed at the teletypewriter. 19a

o/utput/ d/evice/ t/eletype/ CA 19a1

When this command is executed, the current NLS file and its partial copy are printed at the terminal. 19b

The file is printed beginning with the statement to which the Control Marker (CM) is currently positioned. To print an entire file, the CM must be positioned to statement 0 of the file. (The term CM is defined in Section 3. ADDRESSES.) 19c

The user may control the format of the output from within the file by using the directives described in Appendix B of this document. Output format may also be controlled by setting the viewspecs discussed in Section 4 of this manual prior to issuing the Output Device command. 19d

EXECUTE FILE VERIFY 20

The execute file verify command causes the system to check for any problems in the current file that would render it unacceptable for processing by NLS (e.g. structural inconsistancy). 20a

e[xecute] f[ile verify] CA 20a1

In response, the system will print: 20b

FILE VERIFY IN PROGRESS 20b1

If no errors are detected, the system will print the NLS character "*". Otherwise, it issues the message: 20c

BAD FILE -- TYPE CA 20c1

In the event of this message, follow the procedure described in Appendix D under the BAD FILE message. 20c2

EXECUTE RESET 21

The execute reset command creates a partial copy that voids the contents of the current file. 21a

e[xecute] r[eset] CA [really ?] CA 21a1

This command is essentially equivalent to deleting plex 1 of a file. 21b

Like the Execute Unlock Command, this command requires an extra terminating CA to decrease the chance of executing this command by mistake. (Should this command be executed by mistake, the Execute Unlock command may be used to restore the original file, but not the partial copy.) 21c

EXECUTE ASSIMILATE 22

The execute assimilate command causes the system to copy all or part or another NLS file and incorporate it with the current file at a specified location in the current file. 22a

```
e[xecute] a[ssimilate at] ADDR CA EMPTY CA [CR]
                                  $u
                                   d
[from file] FILENAME CA [CR]
[structure] s[tatement at] ADDR     CA VIEWSPECS CA
            b[ranch at]
            p[lex at]
            g[roup at] ADDR CA ADDR                          22a1
```

where first ADDR = statement address in current file after which new content is to be inserted. Any valid sequence of statement address elements described in Section 3 may be used here. 22b

EMPTY = the file specified will be inserted at the level indicated by first ADDR. 22c

$u = any number of up specifications: the file specified will be inserted one level up from first ADDR for each u specified. 22d

d = the file specified will be inserted one level down from first ADDR for each d specified. 22e

FILENAME = the name of the file to be opened. 22f

second ADDR = the address in FILENAME from which the structure specified will be copied. Note: groups require that both the beginning and ending addresses of a group be specified. 22g

Section 3. ADDRESSES IN THE NLS SYSTEM

1

INTRODUCTION 2

An address is the location of a statement (and a character position within that statement) within a file. In NLS addresses are expressed as strings of character codes which, upon interpretation by the NLS system, cause a control marker to point to a specific statement and to a specific character position within that statement. 2a

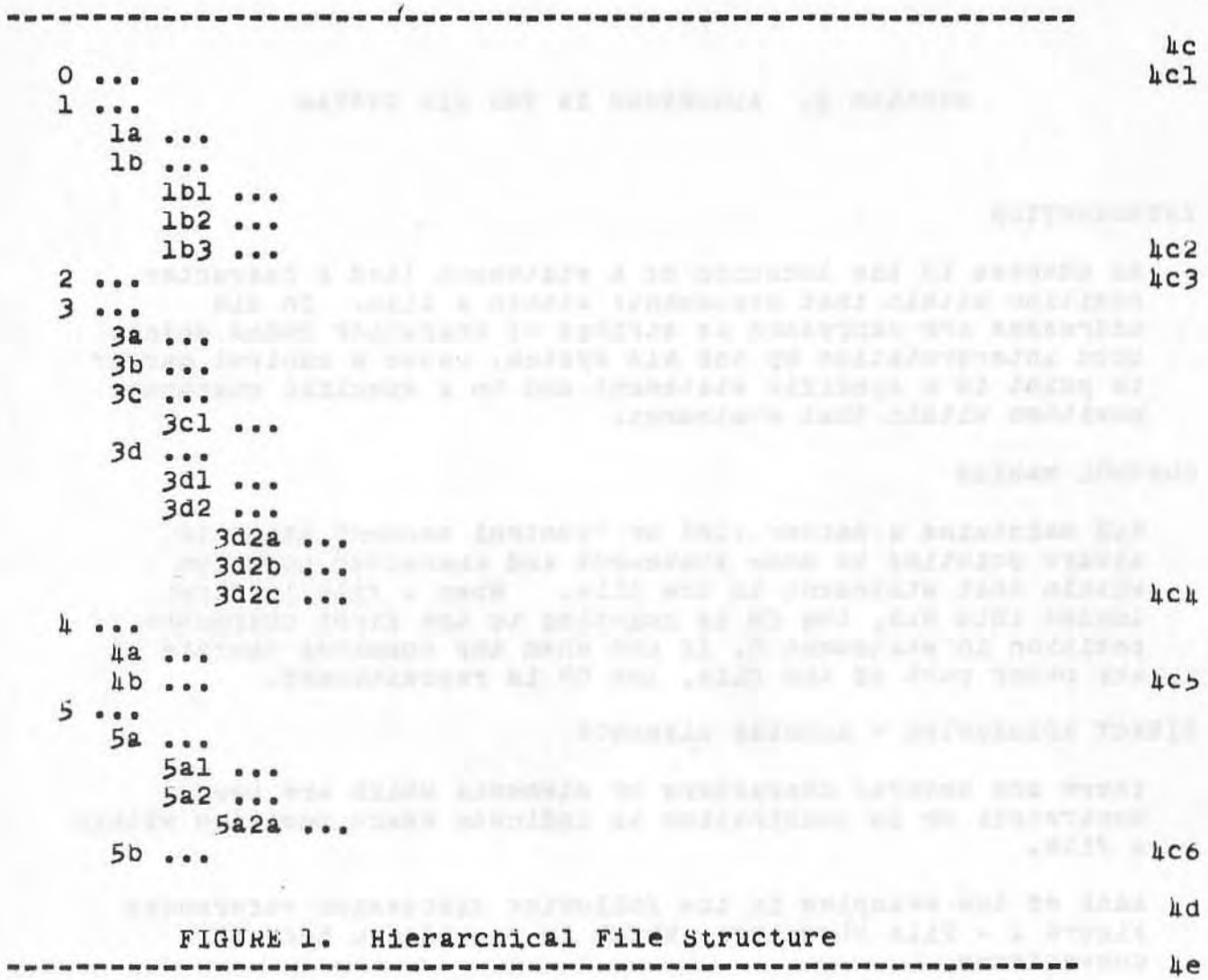
CONTROL MARKER 3

NLS maintains a marker (CM) or "control marker" which is always pointing to some statement and character position within that statement in the file. When a file is first loaded into NLS, the CM is pointing to the first character position in statement 0. If and when any commands operate on any other part of the file, the CM is repositioned. 3a

DIRECT ADDRESSING - ADDRESS ELEMENTS 4

There are several characters or elements which are used separately or in combination to indicate exact position within a file. 4a

Each of the examples in the following discussion references Figure 1 - File Structure, which is reproduced here for convenience. 4b


4c
0 ...
4c1
1 ...
1a ...
1b ...
1b1 ...
1b2 ...
1b3 ...
4c2
2 ...
4c3
3 ...
3a ...
3b ...
3c ...
3c1 ...
3d ...
3d1 ...
3d2 ...
3d2a ...
3d2b ...
3d2c ...
4c4
4 ...
4a ...
4b ...
4c5
5 ...
5a ...
5a1 ...
5a2 ...
5a2a ...
5b ...
4c6
4d
FIGURE 1. Hierarchical File Structure
4e

STATEMENT NUMBERS 4f

A statement number is a series of fields which contain alternately letters and digits. The first field always contains a numeric value. 4g

The total number of fields indicates the level of the statement: 4h

```
1       (1st level)
1a      (2nd level)
1a1     (3rd level)
```
4h1

Statement numbers may be modified by the other elements discussed in this section. A complete set of examples on address specification is included later. 4i

Whenever statement numbers are used in an address specification, they must be preceded by a period (.). 4j

STATEMENT NAMES 4k

Statements may be also referenced by names. Statement names consist of a string of characters; they are enclosed in parentheses and precede all other printing characters of the statement content which they name. 4l

Statement names may include any alphanumeric characters except a right parenthesis. The first character of a statement name must be a letter. 4l1

Statement names are used in address specifications in the same way as statement numbers. They must be preceded by a period. However, unlike statement numbers, they do not have to be unique; the same statement name may be used for multiple statements. The parentheses must be omitted when statement names are used to specifiy address. Example: 4m

```
100 (A) Now is the time for...
```
4m1

Statement 100 may be referenced by .A 4m2

The user may cause the search for the specified statement name to begin from the first statement in the file by using the "F" specification after the statement name. ("F" may be specified in uppercase or lowercase.) For example: 4n

.sam f indicates that the CM should be moved to the first occurrence of a statement named "sam" in the current file. 4n1

If the "f" specification is not used the system searches for the next occurrence of the name starting from the current position of the CM. 4o

STRUCRELS 4p

Every statement in a file may be specified in terms of its structural relationship (strucrel) to other statements in the file. Strucrels describe all the possible (primary and secondary) relationships among statements: When a strucrel is used in conjunction with a statement number or name, it must follow the statement or name and be separated by a space to prevent ambiguities. 4q

u (up)
Refers to the statement preceding the current statement that is one level higher. For example, statement 3 may be referenced as statement 3c u. 4q1

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. down) and/or by an integer value indicating the statement n levels up. 4q1a

The origin statement is the ultimate statement up from every other statement in a file. Any u specification from statement 0 refers to statement 0 itself. Thus, excess u specifications are ignored. 4q1b

d (down)
Refers to the statement following the current statement that is one level lower. For example, statement 4a may be referenced as 4 d. 4q2

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. up) and/or by an integer value indicating the statement n levels down. 4q2a

A d specification to a statement having no following statement on a lower level refers to the statement itself. Thus excess d specifications are ignored. 4q2b

p (predecessor)
Refers to the statement preceding the current statement that is the same level and has the same source. For example, statement 3 may be referenced as 4 p. 4q3

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. successor) and/or by an integer value indicating the nth predecessor statement. 4q3a

s (successor)
Refers to the statement immediately following the current statement that is the same level and that has the same source. For example, statement 3b may be referenced as 3a s 4q4

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. predecessor) and/or by an integer value indicating the nth successor statement. 4q4a

n (head)
Refers to the first statement at the same level that has the same source as the current statement. For example, statement 3a is the head of statement 3b as well as of statement 3a itself. 4q5

This specification pertains only to members of the same plex. 4q5a

t (tail)
Refers to the last statement at the same level that has the same source as the current statement. For example, statement 3d2c is the tail of statement 3d2a. 4q6

This specification pertains only to members of the same plex. 4q6a

e (end)
Refers to the last statement (in hierarchical order) in the branch defined by the current statement. For example, statement 1b2 is the tail statement of any statement in branch 1. 4q7

n (next)
Refers to the statement immediately following the current statement regardless of level or of source. For example: Statement 5 may be referenced as statement 4b n; statement 5a2a may be referenced as statement 5a2 n. 4q8

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. back) and/or by an integer value indicating the nth next statement. 4q8a

b (back)
Refers to the statement immediately preceding the current statement regardless of level and source. For example, 4b may be referenced as 5 b. 4q9

This strucrel may be preceded by a minus sign (-) to indicate its opposite (i.e. next) and/or by an integer value indicating the nth statement back. 4q9a

Strucrels may be used in conjunction with each other and with other address elements. When used with statement numbers they must be preceded by blank spaces to prevent ambiguities in address interpretation. 4r

Examples: 4s

The following address specifications refer to this sample file. 4s1

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a (xx)This was the man to whom all things were
   known;
   1b This was the king who knew the countries of
   the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e (xx)He brought us a tale of the days before
   the flood.                                          4s1a

2 He went on a long journey,
   2a Was weary,
      2a1 Worn-out with labor,
   2b And returning engraved on a stone the whole story.  4s1b
```

ADDRESS	REFERENCES STATEMENT	
		4s2
.1 d	1a	
.1d u	1	
.2a1 uu	2	
.2 ddddd	2a1	
.2a t	2b	
.1d h	1a	
.xx d 2n	1c	
.2b b	2a1	
.2 p	1	
.1 s	2	
.xx f u	1	
.1 e	1e	4s3

LITERAL STRINGS (LIT) 4t

The user may address a statement by content as well as by structural location by specifying a character or string of characters (LIT). There are three ways of expressing literal strings for addressing: /LIT/, <LIT> or ;LIT;. 4u

/LIT/
LIT bounded by brackets causes the system to search for a statement containing LIT. The search is begun from the address specification (if any) preceding the LIT specification, or, if no other address is specified, from the character to the right of the the current position of the CM. When LIT is found, the CM points to the last character of the first occurrence of LIT found. For example, if the CM is positioned to statement 1 and the file contains: 4u1

1 a is for able 4u1a

2 b is for baker 4u1b

3 c is for catastrophic 4u1c

And the user specifies /b/ as an address the CM will be positioned to before the 11th character in statement 1. 4u2

1 a is for a<
>ble 4u2a

At this point the CM is on the first character after the <>. 4u3

<LIT>
LIT bounded by anglebrackets causes the system to search for a statement containing LIT not bounded by numbers or letters. The search is begun from the address specification (if any) preceding the LIT specification, or if no other address is specified, starting from the first character to the right the current position of the CM. When LIT is found, the CM points to the last character of the first occurrence of LIT found. For example, if a file is positioned to statement 1 and contains: 4u4

1 a is for able 4u4a

2 b is for baker 4u4b

3 c is for catastrophic 4u4c

If the user specifies <b> as an address, the CM will be moved to the first character of statement 2 even though the letter b occurs in statement 1 since in statement 2, b is not bounded by letters or digits. 4u5

2 <
>b is for baker 4u5a

;LIT; or 'LIT
LIT bounded by semicolons or a single character LIT preceded by an apostrophe (') causes the search to pertain only to the current statement. For example, if a file is positioned to statement 1 and the file contains: 4u6

1 a is for able 4u6a

2 b is for baker 4u6b

3 c is for catastrophic 4u6c

If the user specifies ;k; or 'k as an address, the CM will not be moved and the command not executed because the system does not look for an occurrence of "k" beyond the current statement. 4u7

Any form of the LIT specification may be preceded by an integer value indicating the nth occurrence of the LIT from the current position of the CM within the file or current statement. 4v

The user may cause the system to repeat the search for the next occurrence of the most recent LIT specified simply by pressing the ALT MODE key. 4w

The user may cause the *[LIT]* or <LIT> searches to start at the beginning of the file by using the "f" specification. ("F" may be specified in uppercase or lowercase.) For example: 4x

[for] f indicates that the CM should be moved to the letter "r" of the first occurrence of the string "for" in the file. 4x1

Using the "f" specification with ;LIT; and 'LIT causes the system to search for the first occurrence in the statement. 4y

If the "f" specification is not used with any LIT specification, the search starts with the character to the right of the current location. 4z

Literal strings may be used in conjunction with other address elements. 4a*

Examples: 4aa

The following address specifications refer to this sample file. (Note that in some cases the actual statement referenced depends on previous address specifications. Thus, each address specification is to be taken in context of previous specifications.) 4aa1

1 O Gilgamesh, lord of Kullab, great is thy praise.
1a (xx)Thiswwas the man to whom all things were known;
1b This was the king who knew the countries of the world.
1c He was wise;
1d He saw mysteries and knew secret things;
1e (xx)He brought us a tale of the days before the flood. 4aa1a

2 He went on a long journey,
2a Was weary,
2a1 Worn-out with labor,
2b And returning engraved on a stone the whole story. 4aa1b

ADDRESS	REFERENCES STATEMENT
[in]	1a (xx)This was the man to whom all thi< >ngs...
<wise>	1c He was wis< >e
[as]	2a Wa< >s weary
[as] f	1a (xx)This wa< >s the man to whom ...
.2 *[a]*	2 He went on < >a long journey

4aa2 4aa3

MARKER 4ab

Markers are another form of address specification. For ease of reference the user may assign a marker for any address in a file which may be subsequently used as a name for that address. Markers are always preceded by the pound sign character "#". 4ac

Unlike statement names, markers must be unique within a particular field. 4ac1

Markers are defined by the Fix Marker command. 4ad

f[ix marker named] NAME CA [at] ADDR CA 4ad1

Where NAME = marker name (1 - 5 alphanumeric characters). Marker names must be unique within a file. 4ae

ADDR = location for which marker is defined. 4af

Markers may be moved within the file by using the Fix Marker command and specifying the old marker name with a new address. 4ag

To obtain a list of the current markers for a file the user may execute the Execute Marker List command: 4ah

e[xecute] m[arker] l[ist] 4ah1

The user may delete a marker specification by using the Execute Marker Release command: 4ai

e[xecute] m[arker] r[elease marker named] NAME CA 4ai1

NOTHING 4aj

If the user does not specify an address field in a command, the current value of the CM is used as the address. 4ak

RETURN/AHEAD 4al

The user may specify an address where the CM was previously positioned by using the RETURN or AHEAD specification. NLS keeps a record of the last few (currently five) intrafile addresses accessed (and any viewspecs in effect at that location) by the user. These intrafile addresses may be referenced by return (r) or ahead (a). 4am

r = return
a = ahead 4aml

Either of these specifications may be preceded by a integer value indicating the number of addresses back or ahead. 4an

Excess return/ahead specifications cause the system to wrap around within the ring. For example, if the last five addresses accessed within the file were statements 2,6,8,9, and 11, and the CM is currently positioned to statement 8, an address of "3 a" would cause the CM to be positioned to statement 2. 4ao

LEFT/RIGHT 4ap

The user may specify an exact location within any statement by using the left-right specification: 4aq

direction	quantity	entity
SP	INTEGER	c
+		w
-		i
		v

4aql

where: 4ar

SP space key (equivalent to +)
+ move forward in current statement
- move backward in current statement
c character 4arl

w word (a contiguous string of letters and/or digits bounded by any characters other than letters or digits) 4ar2

i invisible (a contiguous string of spaces, tabs, and/or carriage returns bounded by any characters other than spaces, tabs and/or carriage returns) 4ar3

v visible (any contiguous string of non-blank characters bounded by any characters other than non-blank characters) 4ar4

The left-right specification causes the CM to be positioned within a statement using the following conventions: 4as

A specification not including an entity type (c, w, i, v) is defaulted to c (character). 4as1

Movement is relative to the number of entities indicated by INTEGER. 4as2

After a left-right specification is evaluated, the CM is pointing to the character specified or to the first character of the word, visible, or invisible. 4as3

The left-right specification moves the CM from its current position to a specified entity. Thus, if the CM is pointing to the first character of the first word of a statement, and the user enters a left-right specification of +3w, the CM will be moved to the first character of the fourth word. If the CM were pointing to the first character in the statement, and that character happened to be an invisible, the same specification would move the CM to the third word in the statement. A specification of +3 will cause the CM to point to the third character to the right of the current CM position. 4as4

If INTEGER is not specified, a value of 1 is assumed. Thus -1w is equivalent to -w. 4as5

A left-right specification moves the CM within a single statement. Thus the CM will not be moved backward past the first character or forward past the last character of the statement. 4as6

The left-right specification may be used in conjunction with any other statement elements. 4at

Examples: 4au

The following address specifications refer to this sample file. (Note that in some cases the actual statement referenced depends on previous address specifications. Thus, each address specification is to be taken in context of previous specifications.) 4au1

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a (xx) This was the man to whom all things were
   known;
   1b This was the king who knew the countries of
   the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of the days before
   the flood.                                          4aula

2 He went on a long journey,
   2a Was weary,
      2a1 (xx) Worn-out with labor,
   2b And returning engraved on a stone the whole story.  4aulb

ADDRESS     REFERENCES STATEMENT
-------     ----------                                  4au2

.1 +4       1 O Gi<
                  >lgamesh, lord of...
 +2w        1 O Gilgamesh, lord <
                               >of...
-w          1 O Gilgamesh, <
                           >lord of...
 n 3i        1a (xx)This was the<
                               > man to whom...
.xx 100w     2a1 (xx) Worn-out with <
                                    >labor,            4au3
```

INDIRECT ADDRESSING - LINKS 5

In addition to the addressing capabilities enabled by the address elements described above, there is another NLS feature which permits the user to address file content indirectly through "links". A link is a special string of text, embedded anywhere in a file or typed as an address specification, which contains a reference to another location in the same file or in any other NLS file. Using links to access other files (in the user's own, or in other users' directories) has essentially the same effect as the load file procedure except that it is quicker to use and enables the user to easily return to the file in which the link first appears. Using links also enables the user to embed precise cross-references in a file for subsequent on-line reading. 5a

The syntax of the link is: 5b

(directory,filename,address:viewspec) 5b1

Where directory = the directory associated with filename. 5c

filename = the name of the file to be accessed (i.e the filename field only). If filename is omitted, the system assumes that the link refers to a location in the current file. 5d

address = a statement number or name indicating the exact location in the file to which the CM will be positioned when the link is executed. If address is not specified, the system assumes the origin statement of the file. 5e

viewspecs = a series of view specifications, or format codes which control the way the file will appear when accessed through the link. If not specified, the system uses any viewspecs currently in effect when the link is executed. For a description of available viewspecs, see Section 4, CREATING AND VIEWING TEXT. 5f

If the directory name is omitted and the link has been typed as part of an address specification, the directory to which the user is currently connected is assumed. If the link references part of the current file, the directory under which the file was created is assumed. This convention allows the user to use links in files from other directories in which the directory name field has been omitted. It also allows the user to omit the directory name from links between files within the same directory. However, there is one exception to this convention. If the filename is a number (digits only), NLS first checks an NLS Journal directory to find the current storage location of a Journal document with that number. If no such journal file exists, the system assumes the directory name as described above. 5g

Note: in the syntax of a link specification directory names are not enclosed in anglebrackets and statement numbers/names are not preceded by a period (.). 5h

Capitalization is ignored for the directory, filename, and address fields of the link specification. When items are omitted, their delimiters may be omitted as well. Examples: 5i

(jones,summary,100:wm)
This link specifies a file contained in a directory belonging to "jones", whose file name is "summary". When the link is executed, the CM will be positioned at statement 100 and statement numbers will be visible (viewspec m) as well as all lines and all levels (viewspec w). Any other viewspecs currently in effect will influence the appearance of the file. 5i1

(myfile,:n)
This link specifies a file belonging to the current user, whose name is 'myfile'. When the link is executed, the file will be positioned at statement 0, and statement numbers will be suppressed (viewspec n). Any other viewspecs currently in effect will influence the appearance of the file. 5i2

(200b)
This link specifies statement 200b in the current file. Any viewspecs currently in effect will apply to the appearance of this part of the file when the link is executed. 5i3

Although address and viewspecifications control the way a file appears when a link is first executed, the user may change any of these parameters once he has used the link. 5j

ADDRESS COMMANDS 6

The following commands are used specifically to control and monitor the positioning of the CM. In this section the word "jump" is used with certain commands which enable the user to position the CM to other files (through links) and to reposition the CM to previously accessed locations in the current as well as recently loaded or linked files. 6a

MOVE CM TO ADDRESS COMMAND 6a1

This command causes the system to move the CM to a specific address. 6a2

SP address CA 6a2a

where address = any valid combination of the following address elements: 6a3

- statement number (must be preceded by a period) 6a4
- statement name (must be preceded by a period) 6a5
- strucrel (may be preceded by a minus sign and/or an integer value) 6a6
- LIT (enclosed in <>, [], ;;, or, if single character LIT, ' and may be preceded by an integer value and/or followed by "f") 6a7
- marker (preceded by a #) 6a8
- left-right specification 6a9
- a (jump to ahead, may be preceded by an integer value 1-5) 6a10
- r (jump to return, may be preceded by an integer value 1-5) 6a11
- LINK specification (must be enclosed in parentheses). 6a12

↑ (jump to link, may be preceded by an integer value indicating the nth link in the current statement to the right of the current location of the CM). This address is valid as long as the CM is positioned anywhere within the statement containing the link specification. 6a13

@ (jump to file ahead, may be preceded by an integer value 1-5) 6a14

& (jump to file return, may be preceded by an integer value 1-5) 6a15

← (move to beginning of statement) 6a16

> (move to end of statement) 6a17

/ (print context of CM when this character is reached in an address specification) 6a18

The difference between the jump to ahead/return and jump to file ahead/return commands is that the former pertains only the last locations accessed in the current file, whereas the latter pertains to locations in the most recently accessed files. 6a19

Examples: 6a20

SP .max f s CA 6a20a

causes the CM to be positioned at the first character of the successor statement to the first occurrence in the file of a statement named "max" 6a20a1

SP .2 /any/ CA 6a20b

causes the CM to be positioned to the first occurrence of the string "any" encountered starting from statement 2. When this is executed the CM will be pointing to the character "y" in "any". If this string is not found, the CM is not moved. 6a20b1

SP .2 ;any; CA 6a20c

causes the CM to be positioned to the first occurence of the string "any" in statement 2 only. If this string is not found, the CM is not moved. 6a20c1

SP (smith,ffile,:x) 6a20d

causes the CM to be positioned to statement 0 of a file named "ffile" in the directory "smith", under control of viewspec x (and any other viewspecs currently in effect). 6a20d1

SP 3r CA 6a20e

causes the CM to be repositioned to the third most recent address accessed in the current file 6a20e1

SP ↑ CA 6a20f

causes the system to position the CM to the location indicated by the first link specification found in the current statement. 6a20f1

PRINT CURRENT CM LOCATION COMMAND 6a21

This command causes the system to print out the current location of the CM: 6a22

. 6a22a

The position of the CM is expressed as a statement number followed by a character position within that statement enclosed in parentheses. 6a23

This command does not require a terminating Command Accept. 6a24

Example: 6a25

.*[=1a(27)]* indicates that the CM is positioned to the 27th character of statement 1a

6a25a

PRINT STATEMENT AT CM COMMAND 6a26

This command causes the system to move the CM to the beginning of the statement at which the CM is currently positioned and to print the statement at the terminal. 6a27

\ 6a27a

There is another version of the slash command which causes the system to print a few characters on either side of the CM with anglebrackets and a line feed break showing character position: 6a28

/ 6a28a

For example if the CM were positioned at the seventh character in statement 1A which contained a series of digits separated by spaces (1 2 3 4 ...), using the slash command would cause the following to be printed: 6a29

```
1 2 3 <
      >4 ...
```
6a29a

These commands do not require a terminating Command Accept.

6a30

PRINT STATEMENT BACK FROM CM COMMAND 6a31

This causes the system to move the CM to the statement which immediately precedes the statement to which the CM currently points and to print the statement at the terminal. 6a32

↑ 6a32a

This command does not require a terminating Command Accept.

6a33

PRINT STATEMENT NEXT TO CM COMMAND 6a34

This command causes the system to move the CM to the statement which is next (one below) to the statement at which the CM is currently positioned according to the current viewspecs and to print the statement at the terminal. 6a35

LF 6a35a

This command does not require a terminating Command Accept. 6a36

Section 4. CREATING AND VIEWING TEXT

1

ENTERING TEXT IN NLS - INSERT COMMAND 2

Text is created in the NLS system using the Insert Command to either enter new statements, or to add text to an existing statement. For purposes of clarity, these two applications of the Insert Command are discussed separately. First, to use the Insert Command to create a new statement: 2a

i[nsert] s[tatement at] ADDR CA EMPTY CA [CR]
$u SP
a

LIT CA
CDOT 2a1

where ADDR = any valid combination of address elements indicating a statement location at which (i.e. after which) the new statement is to be inserted. Following ADDR the user may specify a level adjust (LEVADJ) which determines the level of the new statement. 2b

EMPTY = the statement to be inserted is the same level as ADDR. 2c

$u = (LEVADJ) any number of up level specifications which indicates that the statement to is be inserted x levels higher than ADDR. The number of u elements allowed ranges from 1 to (the level of ADDR). u may also be preceded by an integer value indicating the number of levels up. This specification may include d's . which cancel out u's on a one-to-one basis. 2d

d = (LEVADJ) a down level specification which indicates that the statement will be inserted one level lower than ADDR. This specification may include u's which cancel out d's on a one-to-one basis. 2e

LIT = any series of characters except CA or Centerdot (CDOT) which is the text of the statement to be inserted. The editing characters ↑A (backspace character). and ↑W (backspace word) may be used when entering LIT to correct entries; ↑R maybe used to cause the system to print out the current content of LIT. If LIT is omitted, an empty (blank) line is generated. 2f

CDOT = "center dot" character means continue insert command. This option allows the user to continue inserting statements at the same and/or other levels. When this delimiter is used, the syntax for inserting subsequent statements is the same as though the user had typed the Insert command up to and including the first CA; the system expects the user to enter a level specification and/or LIT. 2g

After this command is executed the CM is positioned to the first character of the most recently inserted statement. 2n

When a new statement is inserted into a file, all statements following the place of insertion are automatically renumbered by the system as necessary. 2i

The maximum number of characters allowed per statement is approximately 2000. Every statement consists of at least one character. 2j

Examples: 2k

If the CM is positioned at statement 3: 2k1

```
i s CA SP
you are my sunshine, my only sunshine CA
```
2k1a

causes the system to insert "you are my ..." into the file as statememt 4. 2k2

```
i s .3 CA d SP
play it again CA                                    2k2a
```

causes the system to insert "play it ..." as statement 3a. 2k3

```
i s .0 [sunshine] CA d SP
you make me happy when skies are grey CA            2k3a
```

causes the system the insert the text "you make me..." after and one level down from the first statement after statement 0 which contains the text sunshine, i.e. after statement 4. The new statement is 4a. 2k4

The second use of the Insert Command is to insert words, characters, text, etc. into an existing statement: 21

```
i[nsert] c[haracter at] ADDR     CA [CR]
         w[ord at]               SP
         v[isible at]
         i[nvisible at]
         n[umber at]
         l[ink at]
         t[ext at]
   LIT   CA
         CDOT                                        211
```

where ADDR = any valid combination of address elements indicating a statement location in the current file and a character position within that statement at which (i.e. after which) the new entity is to be inserted. 2m

LIT = any series of characters except CA or Centerdot (CDOT) which is the text of the entity to be inserted. The editing characters Backspace Character (BC), and Backspace Word (BW) may be used when entering LIT to correct entries; ↑R may be used to cause the system to print out the current content of LIT. If LIT is omitted, the original statement is not modified. 2n

CDOT = continue insert command. This option allows the user to continue inserting entities. When this delimiter is used the syntax for inserting subsequent entities is the same as though the user had typed the insert command; the system expects the user to enter a new address. 2o

After this command is executed, the CM remains positioned to the character specified by ADDR. 2p

The entity specified in this command is significant only in that it affects where and how LIT will be inserted in the statement. Characters and text are treated in exactly the same way; LIT is inserted after the character to which the CM is pointing. Words, visibles, etc. are inserted after the corresponding entity to which the CM is pointing and LIT is inserted with appropriate spacing. 2q

When inserting characters, words, etc. into a statement the user should reference a character position within the statement by using the left/right or LIT address specification. Example: 2r

```
5 Now is the time for all good
i w .5+3w CA [CR]
table CA
(the contents of statement 5 is now:)
5 Now is the time table for all good
```

When the user specifies ".5+3w" as an address in the insert command, the CM is positioned to the third word from where the CM is currently positioned (i.e. to the word "time" since the CM was positioned at the first word in statement 5). When this command is executed the word "table" is inserted after the word at which the CM is currently positioned (i.e. after "time"). 2r1

Remember that when positioning the CM to within a statement from the first character position of the statement, the CM is always moved from the first character in the statement. Thus a specification of +3w references the fourth word if the CM is on the first word and a specification of +2 references the third character. 2s

In the following table, the statement "i.o.u. at least $25.00" is edited using the insert statement and a variety of entities. In each case, the LIT to be inserted is the same (xxx), and each addresss specification assumes that the CM is positioned at the beginning of the statement. 2t

entity	ADDR	new statement	2t1
character	3c	i.o.xxxu. at least $25.00.	
	i	i.o.u. xxxat least $25.00.	
	w	i.oxxx.u. at least $25.00.	
word	12c	i.o.u. at least xxx $25.00.	
	2i	i.o.u. at least xxx $25.00.	
	5w	i.o.u. at least $25 xxx.00.	

visible (same as character)
invisible (same as character)
number (same as word)
link (same as word)
text (same as character) 2t2

When using the Insert Link command, if LIT is not enclosed in parentheses, the text will be inserted as a word, but the following message will be printed: 2u

ILLEGAL LINK 2u1

DISPLAYING TEXT AT THE TERMINAL - PRINT COMMAND 3

The Print command allows Statement, Branch, Group, or Plex as an operand type. It causes the specified part of the file to be printed out on the terminal, with format control by VIEWSPEC parameters (see the discussion of viewspecs in the latter part of this section). 3a

p[rint] s[tatement] ADDR CA VIEWSPEC CA
b[ranch]
p[lex]
g[roup] ADDR CA ADDR 3a1

where ADDR = any valid combination of address elements which specify statement location. If not specified, the current value of the CM is used. 3b

VIEWSPEC = a string of viewspec parameters which affect the way that the output item appears when displayed. Viewspec parameters are described in the latter part of this section. If no viewspecs are specified in this command, the system uses any viewspecs in effect when this command is executed. 3c

After the user presses the final CA, the specified statement, branch, plex, or group is printed at the terminal. The printout can be stopped at any time by pressing the control o key (↑o) which causes printing to cease at the end of the current statement. The user may interrupt printing by using ↑s which causes printing to cease on the current statement and begin again at the next statement to be output. 3d

After the print command is executed, the CM is positioned to the first character position of the last statement output. 3e

The user may specify the whole file including statement 0 to be output with the command "print branch .0"; or the whole file excluding statement 0 with the command "print plex .1". 3f

VIEWSPECS 4

The operation of the Print, Substitute, Execute Assimilate, and Output Device commands is affected by a set of parameters called "viewspecs" (a name derived from display NLS usage, where the same parameters affect the "view" shown on the display screen). 4a

In the NLS Print command viewspecs affect the printout. In the Output Device Teletype command, which is used for formatted hard-copy output, some of the viewspecs affect the output. In the Substitute command, some of the viewspecs determine what portions of the file are influenced by the substitute. 4b

Generally speaking, the most common and important use of viewspecs is to cause some of the statements in the file (or part of the file) to be ignored for various reasons. Thus, for example, certain important viewspecs have the effect of ignoring all statements that are below a specified level in the hierarchical file structure. 4c

When the user first enters NLS, all of the viewspecs are automatically preset to standard values. Whenever the user gives a Print or viewspecs command, he has the option of changing any of the viewspecs by typing special one-letter codes. 4c1

There are two types of viewspecs. The first type includes the Level and Line specifications whose value may range from 1 to ALL. All other viewspecs (the second type) are essentially switches which activate/deactivate various NLS features affecting format. 4d

THE LEVELS VIEWSPEC 4e

The levels viewspec specifies how many levels of the file structure are to be used. Initially, level is set to its standard value of ALL. 4e1

On any Print command except Print Statement, NLS prints only statements whose level is equal to or higher than the current level specification. (The Print Statement command is not affected by the current level specification.) This viewspec also affects the output in the Output Device command and restricts the effect of the Substitute command. 4e2

d sets L to 1
c sets L to ALL
a sets L to L-1
b sets L to L+1
e sets L relative (i.e. L is set to the level of the first statement to be printed by the command, i.e. the statement specified in the command.) For example, if a print statement specified an address of ".5a2", only first, second, and third level statements would be printed. 4e3

where L = current level specification 4e4

THE LINES VIEWSPEC 4f

The lines viewspec is a value from 1 to ALL which allows the user to specify how many lines of each statement are to be printed. The lines viewspec is preset to ALL; if the user changes it to, for example, 3, only the first three lines of any statement will be printed. 4f1

The codes for setting the lines viewspec are as follows: 4f2

t sets T to 1
s sets T to ALL
q sets T to T-1
r sets T to T+1 4f2a

LINES AND LEVELS VIEWSPECS 4g

In addition, to the viewspecs for lines and levels there are two extremely useful codes that affect both levels and lines: 4g1

x sets levels and lines to 1
w sets levels and lines to ALL 4g1a

OTHER VIEWSPECS 4h

The remaining viewspecs are ON/OFF switches for various NLS features. Each is controlled by a pair of one-letter codes, one of which turns the feature ON and the other of which turns it OFF. Note that some of these codes are capital letters; it is important to distinguish between capital and lower-case viewspec codes, because they have different effects. 4h1

PRINTOUT OF STATEMENT NUMBERS ON/OFF (Codes m/n) 4h1a

Normally, when a statement is printed, its statement number is printed at the beginning of the first line. This may be suppressed by the viewspec "n". 4h1b

m turns statement numbers ON
n turns them OFF. 4h1b1

The standard setting for this viewspec is OFF (n). 4h1c

PRINTOUT OF STATEMENT NAMES ON 4h1d

Normally, when a statement is printed, its statement name (if any) is suppressed. Statement names are printed at the beginning each the statement having a name by the viewspec "c". 4h1e

C turns statement names ON
D turns them OFF 4h1e1

The standard setting for this viewspec is ON (C). 4h1f

BLANK LINES BETWEEN STATEMENTS ON/OFF (Codes y/z) 4h1g

The viewspec code "y" causes NLS to put blank lines between statements on output. This makes the printout more legible, especially if statement numbers have been turned OFF. 4h1h

y turns blank lines ON
z turns them OFF. 4h1h1

The standard setting for this device is OFF (z). 4h1i

INDENTATION OF STATEMENTS ACCORDING TO LEVEL ON/OFF
(Codes A/B) 4h1j

NLS normally indents according to level when it prints statements. This can be suppressed by the viewspec "B", causing all statements to be printed flush at the left margin. 4h1k

A turns indenting ON
B turns indenting OFF 4h1k1

The standard setting for this device is ON (A). 4h11

VIEWSPECS COMMAND 4i

The Viewspec command enables the user to use the viewspec features of NLS at any time (i.e. besides during print, link, output device, and substitute operations). 4i1

v[iewspecs] VIEWSPECS CA 4i1a

where VIEWSPECS = any series of valid viewspec codes 4i2

If the user has set viewspecs using this command and subsequently issues a print, output device, or substitute command without any viewspec specification, the viewspecs activated by the viewspec command will affect the operation accordingly. For example, if the user issues the following series of commands: 4i3

```
v m CA
p b .O CA CA
```
4i3a

the current file will be printed with statement numbers. Similarly, if the user had entered the series of commands: 4i4

```
v m CA
p b .O CA n CA
```
4i4a

the current file would be printed without statement numbers as the viewspecs specification in the print statement changes the setting established by the Viewspec command. 4i5

Viewspecs activated by the viewspec, jump to link, and/or print commands remain in effect until deactivated by their opposites in subsequent viewspecs, jump to link, and/or print commands, or until the user leaves NLS. 4i6

VIEWCHANGE 5

The viewchange system is a subcommand mode of NLS which has an extensive set of commands which apply to the appearance of text at the terminal and control certain character assignments, formatting, and feedback mechanisms. 5a

The viewchange system is accessed by the execute viewchange command. 5b

e[xecute] v[iewchange CR] 5b1

Once the user has entered this command he may perform four types of tasks: control character assignment, shift character definition, feedback, and text control. 5c

CONTROL CHARACTER ASSIGNMENT 5c1

The user may name any character from his device's character set to be any control character, i.e., Command Accept (CA), Command Delete (CD), Backspace Character (BC), Backspace Word (BW), Literal Escape, TAB, etc. 5c2

e[xecute] v[iewchange CR]
c[haracter set CR]
d[efine] C1 [as] CC [echo as] C2 CA 5c2a

Where C1 = character to be defined as control character CC 5c3

CC = control character which will be used whenever C1 is typed by the user. CC may be entered by pressing the control character itself, or by typing the name of the control character preceded by a space. For example:
d[efine] x[as] ↑d [echo as] x
is equivalent to:
d[efine] x [as] "SP CA" [echo as] x
The name of the control character may be entered in upper or lower case. 5c4

C2 = the character which will appear at the terminal whenever C1 is entered by the user 5c5

When this command is executed C1 and CC are equivalent, i.e., the user may still type CC as a control character. 5c6

Example:
e/xecute/ v/iewchange CR/
c/haracter set CR/
d/efine/ - /as/ CA /echo as/ - CA 5c6a

Using a minus sign (-) instead of a Command Accept saves the user at a Model 33 teletypewriter terminal, for example, from having to use the control key whenever CA is needed. 5c6b

After the user makes an assignment, he may exit from the viewchange system by typing two successive Command Accepts, execute another viewchange task by typing one Command Accept and specifying the task, or continue defining control characters simply by typing "d" as before. 5c7

Example:
e/xecute/ v/iewchange CR/
c/haracter set CR/
d/efine/ - /as/ CA /echo as/ - CA /CR/
d/efine/ x /as/ CA /echo as/ x CA /CR/
CA /CR/
s/hift case.....
c/haracter set CR/
d/efine/ //as/ BC /echo as/ ↑CA /CR/
CA /CR/
CA /CR/
* 5c7a

In this example, the user may have specified the character "-" instead of CA anywhere after the first character definition task. 5c8

The assignment(s) made by this command remains in effect as long as the user is in NLS. The only way it can be deactivated is to exit NLS, Logout or Reset in EXEC and return to NLS. 5c9

For a list of character set equivalences between the various terminals that use NLS, see Part 4 of Appendix A. 5c10

SHIFT CASE ASSIGNMENT 5c11

The user may name any character from his device's character set to be a shift case character. 5c12

```
e[xecute] v[iewchange  CR]
   s[hift case CR]
      l[ower case]      c[haracter]   CHARACTER  CA
      u[pper case]      w[ord]
      c[ontrol case]    p[ermanent]
      o[ff]
```
5c12a

Where l = the entity (character, word or permanent) specified will be set to lower case whenever preceded by CHARACTER on input. 5c13

u = the entity (character, word or permanent) specified will be set to upper case whenever preceded by CHARACTER on input. 5c14

first c = (control case) The entity (character, word or permanent) specified will be set to control case whenever preceded the by character C on input. This setting is like setting C to be the control key, except that C must precede the subsequent entry instead of being used simultaneously with the subsequent entry. 5c15

second c = CHARACTER will affect the subsequent character entered. 5c16

w = CHARACTER will affect the subsequent word entered (and not intervening spaces). 5c17

p = CHARACTER will affect all subsequent entries until the user deactivates this setting or leaves NLS. 5c18

CHARACTER = the character being defined as a shift/control key 5c19

o = deactivate permanent shift enabled. 5c20

After the user makes an assignment, he may exit from the Viewchange system by typing two successive Command Accepts, execute another viewchange task by typing one CA and specifying the task, or continue defining shift characters simply by typing ""1", "u", or "c"" as before. 5c21

Example: 5c22

```
e[xecute v[iewchange  CR]
s[hift case  CR]
      u[pper case] w[ord]  1 CA [CR]
      u[pper case] p[ermanent] 2 CA [CR]
      CA [CR]
   c[haracter set]...
   s[hift case  CR]
      c[ontrol] c[haracter] q CA [CR]
      CA [CR]
   CA [CR]
*
```
5c22a

The assignment(s) made by this command remain in effect as long as the user is in NLS. The only way it can be deactivated is to exit NLS, Logout or Reset in EXEC and return to NLS. 5c23

For a list of character set equivalences between the various terminals that use NLS, see Part 4 of Appendix A. 5c24

FEEDBACK DEFINED 5c25

The feedback defined command enables the user to specify the number of characters to be echoed for each command word at the terminal and the number of spaces that are indented for each level statement, and to cause the system to echo the statement number and level of a statement to be inserted into a file. 5c26

```
e[xecute] v[iewchange CR]
   f[eedback CR]
      c[haracters xx] XX   CA
      i[ndenting  yy] YY
      l[evadj numbers]
```
5c26a

where xx = the current number of characters echoed per command at the terminal 5c27

XX = a numeric value specifying a setting for xx. If no value is entered, the current value of xx is assumed. 5c28

yy = the current number of spaces indented before text is displayed at the terminal. 5c29

YY = a numeric value specifying a setting for yy. If no value is specified, the current value of yy is assumed.

5c30

TEXT AREA DEFINITION 5c31

The tasks available under this command of the viewchange system enable the user to define tab stops along the print line, specify indenting between successive statement levels, specify the total number of lines per page, and specify the total number of character positions permitted in each line. 5c32

```
e[xecute[ v[iewchange  CR]
   t[ext area  CR]
     t[abs: A [aa] A        CA [CR]
     i[ndenting bb] BB
     l[ines/page cc] CC
     r[ows/page dd] DD
     c[olumns=ee] EE
```
5c32a

where A = a numeric value from zero to nine specifying the first, second, ...tenth tab stop. When this value is entered by the user, the system echoes the current value for that tab stop. 5c33

aa = character position at which the tab stop specified (w) is currently set. 5c34

AA = a numeric value specifying a setting for ww. If no value is specified, the current value of aa is assumed. 5c35

bb = the current number of spaces indented between successive statement levels. 5c36

BB = a numeric value specifying a setting for bb. If no value is specified, the current value of bb is assumed. 5c37

cc = the current number of lines which constitute a page (i.e overall page size). 5c38

CC = a numeric value specifying a setting for cc. If no value is specified, the current value of cc is assumed. 5c39

dd = the current number of lines which may be printed on a page. 5c40

DD = a numeric value specifying a setting for dd. If no value is specified, the current value of dd is assumed. 5c41

ee = current number of characters positions (columns) across each line. 5c42

EE = a numeric value specifying a setting for ee. If no value is specified, the current value of ee is assumed. 5c43

5c44

NOTE: page in this context refers not to an output page, but to the way text appears in "pages" at the terminal when the print command is executed. 5c45

Section 5. TEXT EDITING

1

INTRODUCTION 2

The following commands are used to change the contents of statements and to change file structure. One difference exists in the operation of these commands with respect to whether the operand is a structural entity (statement, branch, etc.) or textual entity (character, word, etc.) - the position of the Control Marker after the command is executed. Commands which operate on structural entities generally cause the CM to be positioned to the first character in the last highest level statement in the structural entity affected by the command. For example, a move operation on a plex causes the CM to be positioned at the source statement of the last branch in the plex. Commands which operate on textual entities cause the CM to remain on the character specified by the destination address. 2a

MOVE COMMAND 3

The move command enables the user to move a statement, branch, plex, or group of statements from one location to another within a file or to move an entity within a statement to another location within the same or another statement. 3a

```
m[ove]  s[tatement to]  ADDR CA [from] ADDR        CA   EMPTY  CA
        b[ranch to]                                     $u     CDOT
        p[lex to]                                       a
        g[roup to]                      ADDR CA ADDR
-------------------------------------------------------------------
        w[ord to]  ADDR CA [from] ADDR             CA
        c[haracter to]                             CDOT
        v[isible to]
        i[nvisible to]
        n[umber to]
        l[ink to]
        t[ext to]                ADDR CA ADDR
```

3b

Where first ADDR = address at which (i.e. after which) the entity specified will be inserted 3c

second ADDR = address of the entity to be moved. In the case of group and text, the beginning and ending address of the group/text must both be specified. 3d

EMPTY = the structural entity to be moved will be inserted at the same level as the first ADDR. 3e

$u = any number of u's indicating that the entity to be moved will be inserted one level higher than the first ADDR for each u specified. u may be preceded by an integer value indicating the number of levels up. u's may be combined with d's and cancel out each other on a one-to-one basis. 3f

d = the structural entity to be moved will be inserted one level lower. than ADDR. 3g

CDOT = continue move command. This option allows the user to continue moving entities (and for structural, entities at the same and/or other levels). When this delimiter is used, the syntax for moving subsequent structural entities is the same as though the user had typed the move command up to and including the first CA; the system expects a new second ADDR to be entered by the user. The syntax for moving textual entities is the same as though the user had typed only the name of the command; the system expects a new first ADDR to be entered by the user. 3h

After this command is executed on structural entities the CM is positioned to the first character of the last, highest level statement moved. After execution on text entities, the CM remains on the character indicated by first ADDR. 3i

After this command is executed, only one copy of the entity exists at the first (destination) address specified. 3j

If in the case of the move statement command, the statement to be moved has any substatements, the command is illegal. NLS aborts the command, prints an error message, and awaits a new command. 3k

It is also illegal to move a structure into itself. 31

Example: 311

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were
   known;
   1b This was the king who knew the countries
   of the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of before the days
   the flood.                                          311a

m w [to] .1e 'y CA [from] ;ef; CA                      311b

/                                                      311c

1e  the da<
          >ys before the                               311d

m s [to] .1b CA [from] .1d CA CA                       311e

p g .1b CA .1d CA CA                                   311f

1b This was the king who knew the countries
of the world.
1c He saw mysteries and knew secret things;
1d He was wise;                                        311g

m s  [to] .1e CA [from] .1c CA d CA                    311h

p b u CA CA                                            311i

1d He brought us a tale of the days before
the flood.
   1d1 He was wise;                                    311j
```

COPY COMMAND 4

The copy command enables the user to reproduce an entity within the file at another location. 4a

```
c[opy] s[tatement to] ADDR CA [from] ADDR      CA  EMPTY   CA
       b[ranch to]                                $u     CDOT
       p[lex to]                                   a
       g[roup to]                    ADDR CA ADDR
-------------------------------------------------------------
       w[ord to] ADDR CA [from] ADDR          CA
       c[haracter to]                         CDOT
       v[isible to]
       i[nvisible to]
       n[umber to]
       l[ink to]
       t[ext to]              ADDR CA ADDR
```

4b

4c

where first ADDR = address at which (i.e. after which) the specified entity will be inserted. 4d

second ADDR = address of the entity to be copied. In the case of group and text, the beginning and ending address of the group/text must both be specified. 4e

EMPTY = the structural entity to be copied will be inserted at the same level as the first ADDR. 4f

$u = any number of u's indicating that the structural entity to be copied will be inserted one level higher than the first ADDR for each u specified. u may be preceded by an integer value indicating the number of levels up. u's may be combined with d's and cancel out each other on a one-to-one basis. 4g

d = the structural entity to be copied will be inserted one level lower than the first ADDR. 4h

CDOT = continue copy command. This option allows the user to continue copying entities (and/or structural entities, at the same and/or other levels). When this delimiter is used, the syntax for copying subsequent structural entities is the same as though the user had typed the copy command up to and including the first CA; the system expects a new second ADDR to be entered by the user. The syntax for copying textual entities is the same as though the user had typed only the name of the command; the system expects a new first ADDR to be entered by the user. 4i

After this command is executed on structural entities the CM is positioned to the first character of the last, highest level statement copied. After execution on text entities, the CM remains on the character indicated by first ADDR. 4j

When this command is executed, two versions of the entity exist. 4k

Example: 4l

```
1 0 Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were
   known;
   1b This was the king who knew the countries
   of the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of the days before
   the flood.                                          411

c w .1b 2w CA /from/ .1c 2w CA                         412

\                                                      413

1b This was the wise king who knew the countries
of the world.                                          414
```

DELETE COMMAND 5

The Delete command enables the user to delete an entity from a file. 5a

d/elete/ s/tatement/ ADDR CA /OK?/ CA
b/ranch/ CDOT
p/lex/
g/roup/
p/lex/
g/roup/ ADDR CA ADDR
w/ord/
c/naracter/
v/isible/
i/nvisible/
n/umber/
l/ink/
t/ext/ ADDR CA ADDR 5a1

where ADDR = any valid combination of address elements which specify the entity to be deleted. Note that group and text require both the beginning and ending address of the group/link to be specified. 5b

CDOT = continue delete command. This option allows the user to continue deleting entities. When this delimiter is used, the syntax for deleting subsequent structural or textual entities is the same as though the user had typed only the name of the command and the entity type; the system expects a new ADDR to be entered by the user. 5c

After this command is executed on structural entities the CM is positioned to the first character of the successor of the deleted structure. After execution on text entities, the CM is positioned to the first character following the deleted entity (or to the source if there is no successor). 5d

In the Delete Statement command, if the statement to be deleted has any substatements, the command cannot be executed. NLS aborts the command, prints an error message, and awaits a new command. (The delete branch command may be used to delete a statement and its substatements.) 5e

Example: 5f

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were
   known;
   1b This was the king who knew the countries of
   the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of the days before the
   flood.                                              5f1

d g .1c CA .1e CA [OK?] CA                             5f2

d s  [king] f CA [OK?] CA                              5f3

p b .1 CA CA                                           5f4

1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were known;  5f5

d w .1 CA [OK?] CA                                     5f6

\                                                      5f7

1 Gilgamesh, lord of Kullab, great is thy praise.      5f8

d c .1a  > CA [OK?] CA                                 5f9

\                                                      5f10

1a This was the man to whom all things were known      5f11
```

REPLACE COMMAND 6

The Replace command causes an entity to be replaced with an equivalent entity. 6a

r[eplace] s[tatement at] ADDR CA [by text?] y[es CR] LIT CA
b[ranch at] n[o] ADDR CA
p[lex at]
g[roup at] ADDR CA ADDR ADDR CA ADDR
w[ord at]
c[haracter at]
v[isible at]
i[nvisible at]
n[umber at]
l[ink at]
t[ext at] ADDR CA ADDR ADDR CA ADDR 6b

where first ADDR = any valid combination of address elements which indicates the location in the file of the entity to be replaced. Note that group and text require that both beginning and ending address of the group/text be specified. 6c

y[es] = the entity specified will be replaced by text entered from the terminal (LIT). 6d

LIT = any string of valid characters. 6e

n[o] = the entity specified will be replaced by an equivalent entity already in the file. (That is, words are replaced by words, branches by branches, etc.) 6f

second ADDR = any valid combiation of address elements which specify the location of the replacement entity. Note that group and text require that both beginning and ending address of the replacement group/text be specified.. 6g

CDOT = continue replace command. For structural entities, the user is placed into the insert statement continuation mode so that successive replacement statements may be entered; the system expects a new LIT to be entered. (New second ADDR's are not accepted.) For textual entities, the user may specify a different first ADDR. 6n

After this command is executed on structural entities the CM is positioned to the first character of the last, highest level replacement statement created. After execution on text entities, the CM remains on the character indicated by first ADDR. 6i

An attempt to replace statement 0 by a second ADDR will cause an error condition. 6j

Example: 6k

```
1 0 Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were
   known;
   1b This was the king who knew the countries
   of the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of the days before
   the flood.                                          6k1

r s [at] .1a CA [by text?] n[o] .1c CDOT [CR]
********** CA                                         6k2

p b .1 CA CA                                          6k3

1 0 Gilgamesh, lord of Kullab, great is thy praise.
   1a He was wise;
   1b **********
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of before the days
   the flood.                                          6k4
```

TRANSPOSE COMMAND 7

This command transposes two entities of the same type. 7a

t[ranspose] s[tatement at]
b[ranch at] ADDR CA [and] ADDR CA
p[lex at] CDOT
g[roup at] ADDR CA ADDR ADDR CA ADDR
w[ord at]
c[haracter at]
v[isible at]
i[nvisible at]
n[umber at]
l[ink at]
t[ext at] ADDR CA ADDR ADDR CA ADDR 7a1

where first ADDR = any valid combination of address elements which indicates the location of one of the entities to be transposed. Note that group and text require that both the beginning and ending address of the group/text be specified. 7b

second ADDR = any valid combination of address elements which indicates the location of the other entity to be transposed. Note that group and text require that both the beginning and ending address of the group/text be specified. 7c

CDOT = continue transpose command. This option allows the user to continue transposing entities. When this delimiter is used, the syntax for transposing structural and textual entities is the same as though the user had typed only the name of the command and entity type; the system expects a new first ADDR to be entered by the user. 7d

After this command is executed on structural entities the CM is positioned to the first character of the statement indicated by first ADDR. After execution on text entities, the CM remains on the character indicated by first ADDR. 7e

When transposing statements with substatements, the substatements are not moved with their source. Substructure is transposed by using the branch, plex, or branch form of the transpose command. 7f

An attempt to transpose branch specifying a branch and any part of the same branch as operands causes an error condition. 7g

Example: 7h

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a This was the man to whom all things were
   known;
   1b This was the king who knew the countries
   of the world.
   1c He was wise;
   1d He saw mysteries and knew secret things;
   1e He brought us a tale of the days before
   the flood.
```
7h1

```
t g .1a CA .1b CA [and]  .1c CA .1e CA
```
7h2

```
p b .1 CA CA
```
7h3

```
1 O Gilgamesh, lord of Kullab, great is thy praise.
   1a He was wise;
   1b He saw mysteries and knew secret things;
   1c He brought us a tale of before the days
   the flood.
   1d This was the man to whom all things were
   known;
   1e This was the king who knew the countries
   of the world.
```
7h4

APPEND COMMAND 8

The Append command enables the user to add the text of one statement to the end of another statement. 8a

a[ppend to] ADDR CA [from] ADDR CA EMPTY CA
LIT CLOT 8a1

where first ADDR = destination statement 8b

second ADDR = be added to the end of the statement at first ADDR. 8c

LIT = optional text which is inserted between the statement at first ADDR and the statement at second ADDR. 8d

CLOT = continue append mode. This option allows the user to continue appending statements. When this delimiter is used, the syntax for appending statements is the same as though the user had typed the append command up to and including the first CA. 8e

After this command is executed the CM is positioned to first ADDR. 8f

If the statement specified by second ADDR has any substructure, the substructure is moved as a plex so that it immediately follows at one level lower the statement at first ADDR (and precedes any substructure associated with the original statement at first ADDR). 8g

Example: 8h

1 O Gilgamesh, lord of Kullab, great is thy praise.
1a This was the man to whom all things were known;
1b This was the king who knew the countries of the world
1c He was wise;
1d He saw mysteries and knew secret things;
1 He brought us a tale of the days before the flood 8h1

2 Humpty Dumpty sat on a wall;
2a Humpty Dumpty had a great fall. 8h2

```
a [to] .1c CA [from] .1d CA and CA                                    8h3

p s CA CA                                                             8h4

1c He was wise; and He saw mysteries and knew
secret things;                                                        8h5

a [to] .1 CA [from] .2 CA CA                                          8h6

p p .1 CA CA                                                          8h7

1 O Gilgamesh, lord of Kullab, great is thy praise.Humpty
Dumpty sat on a wall;
   1a Humpty Dumpty had a great fall.
   1b This was the man to whom all things were
   known;
   1c This was the king who knew the countries
   of the world
   1d He was wise;
   1e He saw mysteries and knew secret things;
   1f He brought us a tale of the days before
   the flood                                                          8h8
```

BREAK COMMAND 9

The Break Command enables the user to break any statement at a specified location causing two separate statements. 9a

b[reak statement at] ADDR CA EMPTY CA
$u CDOT
d 9a1

Where ADDR = location within statement where break is to occur. If the user specifies a character position that falls within a visible in the statement, the statement will be broken immediately following that visible. 9b

$u = any number of u's indicating that the statement following the break will be inserted one level higher than ADDR for each u specified. u's may be combined with d's and cancel out each other on a one-to-one basis. 9c

d = the statement following the break will be inserted one level lower than ADDR 9d

CDOT = continue break command. This option allows the user to continue breaking statements. When this delimiter is used, the syntax for breaking subsequent statements is the same as though the user had typed the break command. The system expects a new address to be entered. 9e

After this command is executed, the CM is positioned to the beginning of the new statement created by the break. 9f

Example: 9g

1 O Gilgamesh, lord of Kullab, great is thy praise.
1a This was the man to whom all things were known;
1b This was the king who knew the countries of the world.
1c He was wise;
1d He saw mysteries and knew secret things;
1e He brought us a tale of the days before the flood. 9g1

```
b [at] .1 bi CA d CA                                    9g2

p b .1 CA CA                                            9g3

1 O Gilgamesh, lord of Kullab,
   1a great is thy praise.
   1b This was the man to whom all things were
   known;
   1c This was the king who knew the countries
   of the world.
   1d He was wise;
   1e He saw mysteries and knew secret things;
   1e He brought us a tale of the days before
   the flood.                                           9g4
```

SUBSTITUTE COMMAND 10

The Substitute command is used to automatically substitute a specified text string for all occurrences of another specified string, throughout a specified statement, branch, plex, or group according to the viewspecs in effect when the substitute is executed. 10a

s[ubstitute] s[tatement at] ADDR CA [CR]
b[ranch at]
p[lex at]
g[roup at] ADDR CA ADDR

[text] LIT CA [for] LIT CA [Go?] y[es]
CA
n[o]... 10a1

Where ADDR = any valid combination of address elements which specify statement location. Note that groups require that both the beginning and ending address of the group be specified. 10b

first LIT = any string of valid characters which will replace the characters specified by the second LIT. 10c

second LIT = any string of valid characters in the statements specified by ADDR which will be replaced by the first LIT. (first LIT and second LIT do not have to correspond in length.) 10d

y[es] = (in response to "Go?") only the substitution(s) indicated by LIT are to be executed. CA is equivalent to typing y[es]. 10e

n[o] = other substitutions are to be made in addition to those already specified. In essence this continues the substitute mode and the subsequent syntax begins with the system request "text". 10f

When muptiple pairs are specified in tne substitute command, tne system looks for matches at each character position in the structure specified for occurrences of all pairs. This mode of operation allows the user to for example substitute all occurrences of tne character "A" for "B" and at the same time substitute "B" for "A". 10g

Viewspecs affect tne operation of this command. For example, if the viewspec x (one level, one line) is in effect when a substitute is performed, only the first line of first level statements will be affected. 10h

After this command is executed, the CM is positioned to the cnaracter indicated by ADDR. 10i

Example: 10j

```
1 and on ...and on ... and on...                          10j1

s s .1 CA
[text] * CA [for text] SP CA [Go?] n[o CR]
[text] / CA [for text] ... CA [Go?] CA                    10j2

p s CA CA                                                 10j3

1 *and*on*/and*on*/*and*on/                               10j4
```

XSET COMMAND 11

The Xset command enables the user to change the case of text entities. 11a

x[set] m[ode] c[apital] CA
l[ower]
s[tatement at] ADDR
c[haracter at]
w[ord at]
n[umber at]
v[isible at]
i[nvisible at]
l[ink at]
t[ext at] 11a1

Where m[ode] = the user may specify the case setting to be used when the Xset command is used on text entities. c[apital] causes the case switch to be set to upper case; l[ower] causes the switch to be set to lower case. The setting determined by the Xset Mode command remains in effect until another Xset Mode command is used. 11b

Although this command allows non-alphabetic arguments (numbers, invisibles, etc.), only alphabetic characters are affected by its execution. 11c

Care should be taken when using the Xset Link command as changing the case of viewspec changes the viewspec itself. 11d

Example: 11e

.lO see document (sam,alpha,100:x) 11e1

x[set] m[ode] c[apital] CA 11e2

x[set] l[ink at] .10 CA 11e3

.lO see document (SAM,ALPHA,100:X) 11e4

EXECUTE EDIT COMMAND 12

The Execute Edit command operates on a single statement and is another means of making detailed alterations to the text of the statement by means of a set of special editing characters. 12a

e[xecute] e[dit] ADDR CA *[CR]*
EDIT TEXT CA 12a1

Where ADDR = any valid combination of address elements which specify a statement location. Note that the execute edit command may be used on one statement at a time only. 12b

EDIT TEXT = a string of mixed editing characters and/or literal input. 12c

The Execute Edit process is essentially the creation of a new statement from an old one; when the Execute Edit is terminated with a CA, the old statement is replaced by the new one. 12d

The command works by moving through the old statement from beginning to end, creating the new one from characters copied from the old one and from characters typed in by the user. This process involves the following operations, which are controlled by special editing characters: 12e

COPYING. Characters are copied one at a time or in strings, from the old statement to the new one. The following characters control copying: 12e1

↑f	copies one character
↑u	copies through end of old statement.
↑z	followed by a typed-in character causes characters to be copied up to and including the next occurrence of the typed-in character.
↑o	followed by a typed-in character causes characters to be copied up to but not including the next occurrence of the typed-in character.
CA	causes the remainder of the old statement to be copied to the new one; the new one then replaces the old and the Edit command is terminated. 12e1a

SKIPPING. Characters in the old statement are skipped over, i.e., the current-location pointer is advanced through the old statement while nothing happens to the new one. (The effect of skipping is to delete characters in the old statement.) The following characters control skipping: 12e2

↑s	skips one character
↑g	followed by a typed-in character causes characters to be skipped up to and including the next occurrence of the typed-in character.
↑p	followed by a typed-in character causes characters to be skipped up to but not including the next occurrence of the typed-in character. 12e2a

INSERTION. Characters typed in by the user go into the new statement without replacing the characters in the old one. 12e3

↑e LIT ↑e	The character string (LIT) between the two ↑e's is entered into the new statement without affecting the old statement. 12e3a

BACKSPACING. The new statement is backspaced, i.e., characters in the new statement are deleted singly or in groups, moving backward in the text. 12e4

↑h	backspaces one character in the new line only without affecting the current-location pointer in the old line
↑w	backspaces one word in the new line only without affecting the current-location pointer in the old line
↑q	moves the current-location pointer in both the old and the new to the beginning of the statement. All editing done prior to pressing this key is deleted.
↑n	one-character "restorative" backspace; it deletes the last character in the new statement and moves the current-location pointer back one character in the old statement.
↑r	causes the existing part of the new statement to be echoed (printed) at the terminal to eliminate any confusion caused by backspacing in the new line. 12e4a

Appendix A. SPECIAL CHARACTERS

1

Part 1 of this appendix is a note on the meaning of CONTROL characters and special "named" characters. Part 2 is a quick-reference summary of special characters used in the Execute Edit command, Part 3 is a quick-reference summary of special characters used in literal input, and Part 4 is a quick-reference summary of the special-character assignments for the various terminal devices.

1a

Part 1. CONTROL CHARACTERS AND "NAMED" CHARACTERS

2

Some of the special characters are called CONTROL characters. A CONTROL character is typed by holding down the CONTROL key and typing the character, then releasing the CONTROL key. 2a

Note that on some terminal devices, the CONTROL key is called something else -- see Section 4 of this appendix. 2a1

Throughout this primer, the up arrow (↑) is used as a special notation to indicate a CONTROL character. Thus the notation "↑d" means that the CONTROL key is held down while a "d" is typed. 2a2

Other special characters are referred to by name, such as CA (Command Accept), CDOT (Continue Command mode), BC" (Backspace Character), "BW" (Backspace Word), etc. 2b

The keys assigned to these named characters vary from one terminal device to another. Section 4 of this appendix gives the assignments for these special characters on various devices. (Note that in many cases, the named characters are control characters.) 2b1

Part 2. EXECUTE EDIT CHARACTERS

CHARACTER	FUNCTION
↑h	backspace character
↑d	copy the rest of the old text into the new and terminate
↑e	insert between angle brackets without changing your place in the old text
↑f	copy one character
↑g	skip up through the character typed following ↑g and type %
↑n	backspace one in old and new text
↑o	copy up to following character
↑p	skip up to following typed character and type %
↑q	backspace statement in old and new
↑r	retype line up to this point
↑s	skip one character in old statement and type %
↑z	copy up through following character
↑u	copy through end of old statement
↑w	backspace word in new statement

Part 3 . SUMMARY OF SPECIAL CHARACTERS IN LITERAL INPUT

4

CHARACTER	FUNCTION	
		4a
LIT ESC	Literal Escape character causes the next character to be taken as a normal text character, even if it is a special character.	4b
CR	Carriage Return used in LIT causes the next entry to begin at the start of a new line.	4c
UCCS	Upper Case Character Shift capitalizes the next character typed.	4d
UCWS	Upper Case Word Shift capitalizes the next word entered.	4e
BC	Backspace Character deletes the last character entered and prints an uparrow (↑).	4f
BW	Backspace Word delete the last word entered but not including its preceding invisible. A back arrow is printed (←).	4g
BS	Backspace Statement deletes the entire current LIT and enables the user to start entering the LIT again.	4h
CDOT	Continue Command mode	4i
CA	Command Accept terminates LIT and all NLS commands.	4j
CD	Command Delete aborts current command.	4k

Part 4. CHARACTER SETS OF DEVICES USED BY NLS

5

CRT	TI	EXECU PORT	TTY33/ TTY35	TTY37	NIC 7-BIT OCTAL CODE	NOTES	
SHIFTI	SHIFT	SHIFT	SHIFT*	SHIFT		* not upper-case on letters	5b
SHIFTII	CTRL	CTRL	CTRL	CONTRL			5c
SHIFT LOCK	SHIFT LOCK	SHIFT* LOCK	none	SHIFT LOCK		* do not hold over spaces	5d
ALT	ESC	ESC	ALT MODE	ESC	033		5e
CA	↑d	↑d	↑d	↑d	004	Command Accept	5f
RETURN	CAR RET	CR	RETURN	RETURN	015		5g
TAB	↑i	↑i	↑i	TAB	011		5h
CD	↑x	↑x	↑x	↑x	030	Command Delete	5i
BACK SPACE	↑a/↑h	↑a/↑n	↑a/↑n	BACK SPACE	010	Delete last character	5j
BACK SPACE WORD	↑w	↑w	↑w	↑w	027	Delete last word	5k
↑q	↑q	↑q	↑q	↑q	021	Delete last statement	5l
↑o	↑o	↑o	↑o	↑o	017	(see Print Command)	5m
↑s	↑s	↑s	↑s	↑s	023	(see Print Command)	5n

CRT	T1	EXECU PORT	TTY33/ TTY35	TTY37	NIC 7-BIT OCTAL CODE	NOTES	
↑j	LINE FEED	LF	LINE FEED	LINE SPACE	012		5p
RUBOUT	DEL	RUBOUT	RUBOUT	DELETE	177		5q
None	PAPER ADV	PAPER EJECT	none	PAPER ADVANCE			5r
(.)	↑b	↑b	↑b	↑b	002	Center dot (continue command mode)	5s
↑v	↑v	↑v	↑v	↑v	026	Literal Escape	5t
↑t	↑t	↑t	↑t	↑t	024	System Status	5u
[	[	[	sK	[	133		5v
]	]	]	sM	]	135		5w
/	/	/	//*	/	057	* echoes ///	5x
\	\	\	slsl*	\	134	* echoes \\\	5y
			sL			Uppercase Word TTY33/35 only	5z
			/			Uppercase Character TTY33/35 only	5a*

NOTES: ↑ = hold control key while pressing character key 5aa

s = hold shift key while pressing character key 5ab

Appendix B. OUTPUT PROCESSOR DIRECTIVES

1

This appendix contains a list of output processor directives which the NIC user will find helpful in formatting text output at the teletypewriter terminal. This list represents a small sampling of all the directives available in the NLS system. Those included here are basic tools. For a description of the complete set of directives currently available, refer to the Output Processor User Guide.

1a

CONTENTS

1b

INTRODUCTION 2

An output processor directive is an instruction to the output processor which dictates how a file is to appear in hardcopy printout. Directives may be included in the origin statement of a file, or embedded at any point within the file itself. No distinction is made between directives in the origin statement and any other statement in a file. 2a

The most recent occurrence of a directive overrides any previous occurrence. All directives have the same format: 2b

.directive; 2b1

Capitalization is critical when using directives; they must be specified exactly as shown here; they must be preceded by a period, and followed by a semicolon. 2c

Leading and trailing blank spaces within the directive specification are ignored. 2d

INFORMATION GENERATION DIRECTIVES

3

GENERATE CURRENT DATE

3a

The user may cause the system to generate the text for the current date by using the GD directive: 3a1

GD 3a1a

This directive may be used in a header specification to cause the current date to be printed at the top of output pages. 3a2

No other parameters are required for this directive. 3a3

GENERATE DATE AND TIME

3b

This directive is similar to the date directive except that the current time of day is also generated: 3b1

GDT 3b1a

This directive may be used in a header specification to cause the current date and time to be printed at the top of each page of output. 3b2

No other parameters are required for this directive. 3b3

HEADER

3c

The header directive is used to specify textual information to be printed as a header message for output pages. 3c1

H = "string" 3c1a

where string = any series of characters (excluding the character " and including output processor directives) string must be enclosed in double quotes. 3c2

Example: H = "REPORT TO NIC USER GROUP" 3c3

GENERATE PAGE NUMBER 3d

This directive causes the system to generate and output the current page number at any point in the text. 3d1

GPN 3d1a

No other parameter is required for this directive. 3d2

Page numbers are always printed outside (below) the lower margin established by the user or by system default and normally at the center of the bottom of the page. 3d3

PAGINATION CONTROL DIRECTVES

4

These directives enable the user to control pagination within a file.

4a

PAGINATE AT END OF STATEMENT

4b

Whenever this directive appears in a statement the system will automatically skip to a new page after printing the statement.

4b1

PES

4b1a

No other parameters are required for this directive.

4b2

PAGINATE TO FIT STATEMENTS

4c

This directive causes the system to determine whether each statement will fit in its entirety onto the current page and if not cause the statement to be printed at the beginning of a new page.

4c1

Pfit

4c1a

No other parameters are required for this directive.

4c2

This is a global directive, i.e., it affects all statements within a file.

4c3

GRAB LINES 4d

This directive is similar to the PFit directive except that it enables the user to specify a group of lines. It causes the system to determine whether the specified number of lines (starting with the first line of the statement in which it appears) will fit on the remainder of the current page, and if not cause them to be output starting on the next new page. 4d1

Grab = x 4d1a

where x = a numeric value from 1 to 148 indicating the number of lines to be evaluated for pagination. This value must reflect all non-printing as well as printing lines. 4d2

This directive affects only the statement in which it appears. 4d3

FORMAT CONTROL DIRECTIVES

5

SET BODY LEFT MARGIN 5a

This directive specifies the character position at which any line of text is to begin within the body of the output page (i.e. excluding the header). 5a1

BLM = x 5a1a

where x = a numeric value of from -131 to +131 indicating left margin setting for the body of the page. The default value for this parameter is 0. 5a2

A negative value in this directive is used to suppress indentation. (Considering that each level downwards in a branch is indented 3 character positions from its source statement.) Thus, the user may cause statements that would normally be indented according to level to be placed flush with the left margin by using this directive. For example, if the user wants to position a third level statement so that it begins at the left margin the directive "LM = -6;" may be used. To determine the minus value required to line up any level statemnt to the left margin set x = (level of the statement - 1) times -3 character positions. Excess minus specifications are ignored, e.g., using the directive "LM = -100" on a second level statement is equivalent to using the directive "LM = -3". 5a3

SET BODY RIGHT MARGIN 5b

This directive enables the user to specify the characte position at which any line of text is to end within the body of the output page (excluding the header).. 5b1

BRM = x 5b1a

where x = a numeric value from 1 to 132 indicating the ending character position of the output line. Although the maximum value allowed for this parameters is 132 characters, typical teletype carriage width permits up to 72 characters only. 5b2

SET LEFT HEADER MARGIN 5c

This directive specifies the character position at which header text is to begin on the output page. 5c1

HLM = x 5c1a

where x = a numeric value of from -131 to +131 indicating left margin setting for the header of the page. The default value for this parameter is 0. 5c2

All negative specifications for this directive are interpreted as 0. 5c3

SET RIGHT HEADER MARGIN 5d

This directive specifies the character position at which header text is to end on the output page. 5d1

HRM = x 5d1a

where x = a numeric value from 1 to 132 indicating the header right margin setting on the output line. Although the maximum value allowed for this parameters is 132 characters, typical teletype carriage width permits up to 72 characters only. 5d2

SET BOTTOM MARGIN 5e

This directive enables the user to define the bottom margin for output pages: 5e1

BM = x 5e1a

where x = a numeric value from of 1 to 150 which indicates the number of lines from the top of a page where the bottom margin for output is to be set. 5e2

SET TOP MARGIN 5f

This directive specifies the top margin of the output page, i.e. the line at which output is to begin on the page. 5f1

TM = x 5f1a

where x = a numeric value of from 1 to 148 indicating the line at which output is to begin on the page. 5f2

HALT OUTPUT 5g

This directive causes the processor output to ignore subsequent file content. 5g1

Halt 5g1a

The user can selectively output a file by using this directive in conjunction with the file's CM. For example, the user can output only the second branch of a file by first positioning the CM to branch 2, and then inserting the Halt directive immediately following the last statement in branch 2. 5g2

STATEMENT NUMBERS ON/OFF 5h

This directive may be used to either suppress the printing of statement numbers, or cause them to be output at the beginning of each statement. 5h1

SN = x 5h1a

where x = 0 (OFF) if statement numbers are to be suppressed
1 (ON) if statement numbers are to be printed 5h2

If this directive is not specified, its default value is determined by the user's status in NLS. That is, if the viewspec that causes statement numbers to appear is currently active when the user performs the output operation, SN is considered set to 1, otherwise 0. 5h3

STATEMENT NUMBER FORMAT 5i

This directive enables the user to specify the character position in the output page at which statement numbers are to be printed. 5i1

SNF = x 5i1a

where x = a numeric value of from 0 to 132 which indicates the character position in the output page in which the statement number will appear. 5i2

If SNF = 1, statement number appear to the far left of the page are left-justified. If SNF is set to any other non-zero value, statement numbers are right-justified. SNF = 0 turns off this option. 5i3

This directive is completely independent of the Statement Numbers (SN) directive, and it is possible to have both kinds of statement numbers printed at the same time. 5i4

The default value for this feature is 0. 5i5

NUMBER OF LINES BETWEEN STATEMENTS 5j

This directive causes the sytem to generate the specified number of blank lines between statements. 5j1

LBS = x 5j1a

where x = a numeric value of from 0 to 147 indicating the number of blank lines to be generated. 5j2

The default value for this feature is 0. 5j3

NUMBER OF LINES BETWEEN LINES 5k

This directive specifies the number of blank lines to be generated between adjacent lines of a statement. 5k1

LBL = x 5k1a

where x = a numeric value of from 0 to 149 indicating the number of blanks lines to be generated. 5k2

The default value for this option is 0. 5k3

DIRECTIVE CONTROL 6

PRINT DIRECTIVES 6a

This directive controls the appearance of directives as part of the text of an output file. 6a1

D = x 6a1a

where x = 1 (Yes) if directives are to appear in output
0 (No) if directives are to be suppressed from output 6a2

The default value for this feature is 0 (do not print directives). 6a3

The default value for this directive is 0 (recognize directives). 6a4

IGNORE DIRECTIVES 6b

This directive causes the system to ignore all subsequent directives until the ignore directives directive is deactivated. 6b1

IgD = x 6b1a

where x = 1 (Yes) if directives are to be ignored
0 (No) if directives are to be recognized 6b2

Appendix C. ERROR MESSAGES

?

Cause: This is a general message which is issued whenever a statement address or command cannot be recognized by NLS.
Action: Enter correct command or address.

BAD FILE -- TYPE CA

Cause: The system found an error in the file structure.

Action:
1. Issue the Execute Quit command, enter NLS, and attempt to load the file.
2. Execute File Verify. If still bad continue to next step.
3. Check partial copy file. Issue the Execute Unlock command to delete the current partial copy of the file.
4. Execute File Verify. If still bad continue to next step.
5. If at this point the error message persists for the file, the only recourse is to return to an earlier version of the file. Go to EXEC, delete the current version, reenter NLS and load a previous version of the file.

CANNOT DELETE OLD VERSIONS

Cause: NLS was unsuccessful in deleting an old file version during a file creation command.
Action: The file may be deleted by the user at the EXEC level.

CANNNOT DELETE PARTIAL COPY

Cause: A system file error has occurred during an output or Update file operation.
Action: None. The current file is not affected; the partial copy still exists in the user's directory.

EXCEED CAPACITY

Cause: A text string was created by either the user or NLS which was too long.

Action: Shorten text or treat as NLS system error.

FIELD TOO LARGE

Cause: A filename specified exceeds 39 characters in length.
Action: Modify filename appropriately.

FILE ALREADY OPEN

Cause: The user has attempted to open an already open file.
Action: None.

ILLEGAL APPEND

Cause: The user has attempted to append a statement to itself.

ILLEGAL CHARACTER

Cause: 1. There is a non-formatting control character included somewhere in the text of a statement.
2. A hardware transmission error has ocurred.
Action: 1. Ascertain location of illegal character and delete it.
2. Exit NLS, enter NLS, and try again.

ILLEGAL DELETE

Cause: 1. The user has tried to delete an entity including statement 0.
2. The user has tried to delete a statement having substatements.
Action: 1. None. The origin statement cannot be deleted.
2. Specify branch in the Delete command to delete a statement and its substatements.

ILLEGAL LINK

Cause: A link specification is syntactically incorrect, e.g., parenthesis are missing.
Action: Correct link specification as appropriate.

ILLEGAL MOVE

Cause:
1. The user has attempted to move statement 0.
2. The user has attempted to move a statement having substatements. (This is equivalent to deleting the statement from it original place in the file.)

Action:
1. None. The origin statement cannot be moved/deleted.
2. Specify branch in the Move command to move a statement and its substatements.

ILLEGAL TRANSPOSE

Cause: The user has issued a transpose command that affects statement 0.
Action: None. The origin statement may not be transposed.

IO ERROR

Cause: A system error has occured during a file transfer.
Action:
1. Try again.
2. Delete the most recent version of the file.

NLS SYSTEM ERROR

Cause: An NLS system error has occurred.
Action: Issue an Execute Quit command and reenter NLS.

NO ROOM IN DIRECTORY

Cause: The user has attempted to create a new file for which there is no room in the current directory.
Action: Go to EXEC and Delete/Expunge files no longer needed to make room for new file.

NO SUCH FILE

Cause: The user has specified a non-existent filename in a Load Command.
Action: 1. Use another filename in the load command.
2. Create a file of the name desired by doing an Output file in NLS, or Copy at the EXEC level.

SYSTEM ERROR

Cause: A timesharing system failure has occurred.
Action: 1. Execute Quit and enter NLS again.
2. Logout, and Login again.

filename LOCKED BY username

Cause: The user has loaded a file which has a partial copy created by another user.
Action: 1. Request "username" to unlock file.
2. Copy the file into your directory at the EXEC level and then proceed.
3. Output file to "read" file but not modify it.

Appendix D. COMMAND SUMMARY

Section 1. EXECUTIVE COMMANDS

Section 2. FILE COMMANDS

l[oad] f[ile] FILENAME CA (p.11)

u[pdate file CA
o[(to old version)] CA (p.13)

o[utput] f[ile] FILENAME CA (p.14)

e[xecute] u[nlock] CA [really ?] CA (p.15)

o[utput] d[evice] t[eletype] CA (p.16)

e[xecute] f[ile verify] CA (p.17)

e[xecute] r[eset] CA [really ?] CA (p.18)

e[xecute] a[ssimilate at] ADDR CA EMPTY CA [CR]
$u
d
[from file] FILENAME CA [CR]
[structure] statement [at] ADDR CA VIEWSPEC CA
branch [at]
plex [at]
group [at] ADDR CA ADDR (p.19)

Section 3. ADDRESS COMMANDS

MOVE CM TO ADDRESS COMMAND (p. 19)

SP ADDR CA

where ADDR = statement number

statement name

strucrel (may be preceded by a minus sign and/or an integer value)

LIT

marker

left-right specification

a (jump to ahead, may be preceded by an integer value 1-5)

r (jump to return, may be preceded by an integer value 1-5)

LINK specification (must be enclosed in parentheses)

↑ (jump to link, may be preceded by an integer value indicating the nth link in the current statement to the right of the current location of the CM)

@ (jump to file ahead, may be preceded by an integer value 1-5)

& (jump to file return, may be preceded by an integer value 1-5)

← (move to beginning of statement)

> (move to end of statement)

\ print context

or any valid combination of the above

PRINT CURRENT CM LOCATION COMMAND (p. 22)

.

PRINT STATEMENT AT CM COMMAND (p. 22)

\

/

PRINT STATEMENT BACK FROM CM COMMAND (p. 23)

↑

PRINT STATEMENT NEXT TO CM COMMAND (p. 23)

LF

Section 4. TEXT CREATING & VIEWING COMMANDS

```
i[nsert] s[tatement at] ADDR CA EMPTY    CA    [CR]
                          $u      SP
                           d
LIT  CA
     CDOT                                              (p.1)
-------------------------------------------------------------
i[nsert] c[haracter at] ADDR          CA  [CR]
         w[ord at]
         v[isible at]
         i[nvisible at]
         n[umber at]
         l[ink at]
         t[ext at]
LIT  CA
     CDOT                                              (p.4)

p[rint] s[tatement] ADDR       CA VIEWSPEC CA
        b[ranch]
        p[lex]
        g[roup] ADDR CA ADDR                           (p.7)

v[iewspecs] VIEWSPECS  CA                              (p.12)

e[xecute] v[iewchange CR]                              (p.13)
   c[haracter set CR]
   [define] C1 [as] CC [echo as] C2  CA

e[xecute] v[iewchange  CR]                             (p.15)
   s[hift case CR]
      l[ower case]      c[haracter]   CHARACTER  CA
      u[pper case]      w[ord]
      c[ontrol case]    p[ermanent]
      o[ff]

e[xecute] v[iewchange CR]                              (p.17)
   f[eedback CR]
      c[haracters xx] XX   CA
      i[ndenting  yy] YY
      l[evadj numbers]
```

```
e[xecute[ v[iewchange  CR]                              (p.17)
   t[ext area  CR]
     t[abs: A[aa] AA       CA [CR]
     i[ndenting=bb] BB
     l[ines/page=cc] CC
     r[ows/page=dd] DD
     c[olumns=ee] EE
```

Section 5. TEXT EDITING COMMANDS

```
m[ove]  s[tatement to]  ADDR CA [from]  ADDR       CA   EMPTY  CA
        b[ranch to]                                     $u     CDOT
        p[lex to]                                       d
        g[roup to]                      ADDR CA ADDR
-------------------------------------------------------------------
        w[ord to]  ADDR CA [from]  ADDR          CA
        c[haracter to]                           CDOT
        v[isible to]
        i[nvisible to]
        n[umber to]
        l[ink to]
        t[ext to]                  ADDR CA ADDR          (p.1)

c[opy]  s[tatement to]  ADDR CA [from]  ADDR       CA   EMPTY  CA
        b[ranch to]                                     $u     CDOT
        p[lex to]                                       d
        g[roup to]                      ADDR CA ADDR
-------------------------------------------------------------------
        w[ord to]  ADDR CA [from]  ADDR          CA
        c[haracter to]                           CDOT
        v[isible to]
        i[nvisible to]
        n[umber to]
        l[ink to]
        t[ext to]                  ADDR CA ADDR          (p.4)

d[elete]  s[tatement]              CA [OK?]  CA
          b[ranch]  ADDR                     CDOT
          p[lex]
          g[roup]  ADDR CA ADDR
          w[ord]
          c[haracter]
          v[isible]
          i[nvisible]
          n[umber]
          l[ink]
          t[ext]  ADDR CA ADDR                        (p. 6)
```

e[xecute[v[iewchange CR] (p.17)
t[ext area CR]
t[abs: A[aa] AA CA [CR]
i[ndenting=bb] BB
l[ines/page=cc] CC
r[ows/page=dd] DD
c[olumns=ee] EE

Section 5. TEXT EDITING COMMANDS

```
m[ove]  s[tatement to] ADDR CA [from] ADDR      CA  EMPTY  CA
        b[ranch to]                                $u    CDOT
        p[lex to]                                  d
        g[roup to]                    ADDR CA ADDR
-------------------------------------------------------------
        w[ord to] ADDR CA [from] ADDR          CA
        c[haracter to]                         CDOT
        v[isible to]
        i[nvisible to]
        n[umber to]
        l[ink to]
        t[ext to]                ADDR CA ADDR        (p.1)

c[opy]  s[tatement to] ADDR CA [from] ADDR      CA  EMPTY  CA
        b[ranch to]                                $u    CDOT
        p[lex to]                                  d
        g[roup to]                    ADDR CA ADDR
-------------------------------------------------------------
        w[ord to] ADDR CA [from] ADDR          CA
        c[haracter to]                         CDOT
        v[isible to]
        i[nvisible to]
        n[umber to]
        l[ink to]
        t[ext to]                ADDR CA ADDR        (p.4)

d[elete] s[tatement]              CA [OK?] CA
         b[ranch] ADDR                     CDOT
         p[lex]
         g[roup]  ADDR CA ADDR
         w[ord]
         c[haracter]
         v[isible]
         i[nvisible]
         n[umber]
         l[ink]
         t[ext]  ADDR CA ADDR                    (p. 6)
```

```
r[eplace] s[tatement] ADDR CA [by text?] y[es CR] LIT   CA
          b[ranch]                       n[o] ADDR     CDOT
          p[lex]
          g[roup] ADDR CA ADDR
          w[ord]
          c[haracter]
          v[isible]
          i[nvisible]
          n[umber]
          l[ink]
          t[ext] ADDR CA ADDR                          (p. 8)

t[ranspose] s[tatement at]
            b[ranch at] ADDR      CA [and] ADDR        CA
            p[lex at]                                  CDOT
            g[roup at] ADDR CA ADDR        ADDR CA ADDR
            w[ord at]
            c[haracter at]
            v[isible at]
            i[nvisible at]
            n[umber at]
            l[ink at]
            t[ext at]  ADDR CA ADDR        ADDR CA ADDR
                                                   (p.10)

a[ppend to] ADDR CA [from] ADDR CA  EMPTY    CA
                                    LIT      CDOT (p.12)

b[reak statement at] ADDR CA  EMPTY   CA
                              $u      CDOT
                              d                    (p.14)

s[ubstitute] s[tatement at]             CA [CR]
             b[ranch at] ADDR
             p[lex at]
             g[roup at] ADDR CA ADDR

[text]  LIT CA  [for] LIT CA [Go?] y[es]
                                   CA
                                   n[o]...          (p.16)
```

x[set] m[ode] c[apital] CA
l[ower]
s[tatement at] ADDR
c[naracter at]
w[ord at]
n[umber at]
v[isible at]
i[nvisible at]
l[ink at]
t[ext at] (p.18)

e[xecute] e[dit] ADDR CA [CR]
EDIT TEXT CA (p.19)

GLOSSARY

1

ADDRESS 2

the location of statement (and a character position within that statement) within a file.

2a

BACK 3

the statement immediately preceding the current statement regardless of level and source.

3a

BRANCH 4

an entity that consists of a specified statement and all its substatements, all their substatements, etc.

4a

CONTROL MARKER 5

in TNLS, a "current position marker" which is always pointing to some statement and to a particular character position within that statement.

5a

DIRECTIVE 6

a text entity embedded in a statement which is interpreted as an instruction by the NLS output processor dictating the appearance of a file in hardcopy printout.

6a

DOWN 7

the statement immediately following the current statement that is one level lower.

7a

END 8

the last statement in the branch defined by the specified statement.

8a

GROUP 9

a subset of a plex identified by two branches (which must be in the same plex) consisting of those two branches plus all branches that fall between them in the same plex.

9a

HEAD 10

the first statement at the same level that has the same source.

10a

INVISIBLE 11

a contiguous string of non-printing characters (such as spaces, tabs, and carriage returns).

11a

LEVADJ 12

a sequence of level adjust specifications (u's and d's) used to determine new levels.

12a

LINK 13

a special string of text embedded anywhere in a file or typed in an address which contains a reference to another location in the same file or any other NLS file.

13a

LIT 14

A series of any characters except CA or CDOT.

14a

MARKER 15

a name for a character position in a statement. Markers are prefaced in addresses by the pound sign character (#).

15a

NEXT 16

refers to the statement immediately following the current statement regardless of level or source.

16a

NIC DIALOG SUPPORT SYSTEM 17

a set of procedures which enables the automatic storage, cataloging, and distribution of information items (messages and documents) within the NLS user group.

17a

NLS FILE 18

a file which may be edited and viewed in NLS.

18a

NUMBER 19

a string of digits, which may optionally be preceded by a minus sign and/or dollar sign, may optionally include a decimal point, and may optionally have commas within it.

19a

ORIGIN STATEMENT 20

the "zero level" statement of every file containing status and version information about the file.

20a

PARTIAL COPY 21

a file consisting of only the changes that have been made to its associated NLS file.

21a

PLEX 22

an entity consisting of a specified branch plus all other branches of the same level that have the same source.

22a

PREDECESSOR 23

the statement immediately preceding the referenced statement that is the same level and has the same source.

23a

SOURCE 24

the statement of which the referenced statement is a substatement. 24a

STATEMENT NAME 25

a string of characters enclosed in user-defined delimiters (parentheses by default) which is used as a statement address. 25a

STATEMENT NUMBER 26

a string of alternating fields of letters and numbers. Statement numbers are the primary means of addressing file content in NLS. 26a

STRUCREL 27

a specification of the structural relationship between statements, e.g., up, down, back, source. 27a

SUBSTATEMENT 28

a statement one level down in the same branch as the referenced statement. 28a

SUCCESSOR 29

the statement immediately following the referenced statement that is the same level and has the same source. 29a

TAIL 30

the last statement at the same level as the referenced statement that has the same source. 30a

TEXT 31

a contiguous string of characters within a statement, delimited by two character position addresses. 31a

UP 32

the statement preceding the current statement that is one level higher than the current statement.

32a

VIEWCHANGE SYSTEM 33

a submode of NLS which controls TNLS character assignments, formatting at the terminal, and feedback mechanisms.

33a

VIEWSPECS 34

a set of parameters which affect the way text is displayed at the terminal and printed on hardcopy devices.

34a

VISIBLE 35

a contiguous string of printing characters.

35a

WORD 36

a contiguous string of letters and/or digits. 36a

37

38

39

INDEX

Catalog Production Automaton Post-Mortem Plans

Catalog Production Automaton Post-Mortem Plans

On Thursday, 5 August 1971, RWW, JDH, BLP, CHI, and myself met to discuss the new Collector/Sorter/Merger, the CPA, and related subjects. 1

In the course of this meeting a new approach for implementing the catalog-related processes was adopted by acclamation. This new approach significantly alters the CPA user-interface as well as changing the nature of my role in helping provide a catalog-production facility. 1a

In this memo I will describe the changes agreed upon, as I understand them, so that the other meeting participants can compare notes. 1b

The following technique for debugging the new system features was proposed: 2

Rather than immediately starting to modify NLS so as to provide the new features, we should instead implement these features through L10 programs that can be run as NLS "User Programs" (as in the analyser-formatter). 2a

Entry points to all needed core-NLS routines can be made available to the User Program compiler, so that (almost?) anything which could be done "inside" NLS can be done by a User Program. 2a1

When the new features have been tested and debugged in this form, they can then be completely integrated into NLS (as commands) -- if this seems to be desirable. 2a2

This technique will help us to avoid many of the pitfalls usually involved in integrating new subsystems into a large system like NLS, specifically: 2b

We will not have to inject ourselves into the compile/load/test/debug cycles already going on within the NLS Group and can maintain a clean interface between the new features and the base system. This will save us a considerable amount of time and eliminate a lot of friction. 2b1

We will have freedom to play around with the new features before having to provide polished user interfaces such as NLS commands. This will allow us to gain experience in using the new features which will assist us in designing the final user interface. 2b2

By using L10 as our linguistic base, we can implement (and put into use) the core routines which are needed for performing catalog operations without having to design and implement a new user-interface language. This should mean that these features will be available much sooner than would otherwise be possible. 2b3

In addition, we will be getting a catalog-production system that has all of the basic power of L10, rather than a system with strictly limited capabilities. 2b3a

Using this technique, we arrive at the following breakdown of tasks needed to implement the first catalog-production system: 3

Implement core sort and merge routines (JDH) 3a

These are the basic programs needed to implement a sort/merge process for which all the parameters have been specified -- i.e., Dave does not have to worry about user interaction at this time, since his programs will be called from an L10 user program, and no NLS command language machinery will be needed. 3a1

Implement multiple user-program buffers in NLS (CHI) 3b

It will be necessary to change NLS so that there can be at least two user programs loaded simultaneously -- e.g., the controlling user program must be able to compile an analyser/formatter program into core and then start up other operations which use that program. 3b1

It would even be nice to have more than two user-program buffers so that more than one compiled analyser/formatter program could be available for "instant" use. 3b1a

It should be possible to start execution of these programs at any labelled location so as to provide for restarts following crashes. 3b1b

Provide catalog-production support (WLB) 3c

Adoption of this plan virtually kills the Catalog-Production Automaton task, and I will instead be involved in providing Dick with the support needed to tie together the appropriate NLS and TENEX features into a package that will satisfy the same needs for which the CPA was designed. 3c1

Specific tasks include: 3c2

Making sure that linkages to all the necessary NLS core routines have been established, and testing the various operations individually. 3c2a

Writing any programs that are needed for interfacing Dave's basic sort and merge routines to higher-level function calls. (The collect operation just involves calls on core-NLS Assimilate routines, so no new core routines are needed for this.) 3c2b

Programming and testing the first catalog-production program to be sure that all the pieces fit together well. 3c2c

Possibly adding some new commands to NLS, if we feel that there are well-defined parts of the catalog-production process that deserve command status. Or, the implementation of a catalog-production language, as originally planned, after the underlying procedures have been implemented and tested. 3c2d

<JOURNAL>7483.NLS;1, 11-AUG-71 13:32 WLB ; (Expedite) Title:
Author(s): Walter L. Bass/WLB; Distribution: James C. Norton, Jeanne B. North, Richard W. Watson, J. D. Hopper, Bruce L. Parsley, Charles H. Irby, William H. Paxton, Walter L. Bass, Douglas C. Engelbart/JCN JBN RWW JDH BLP CHI WHP WLB DCE; Keywords: ; Sub-Collections: ARC; Clerk: WLB;
Origin: <BASS>CPA2.NLS;1, 10-AUG-71 21:49 WLB ;

Imlac Configuration Guide

Imlac configuration Guide 1

The basic IMLAC comes with 4K of memory, no interrupt system, and only a teletype for I/O. This document discusses most of the options which the buyer can add on. 2

More core: 3

This only seems necessary if you want complicated pictures or if you want the IMLAC to function as a real computer rather than an intelligent terminal. The IMLAC/NLS software will run in 4K if you can accept most of your teletype simulation display being lost when you go into NLS. 3a

IMLAC will sell you the wiring for additional memory for some small fraction of the memory cost, if you think you might want to expand later: once your IMLAC has arrived, you cannot add more memory unless you ordered the wiring beforehand. 3b

Memory protect: 4

This might be useful to protect a debugger, although if you have a tape cassette you can reload all of core in about 8-9 minutes with the old 30 CPS model or 1/3 of that with the new model. Our experience is that it's not worth it. 4a

Programmer's console: 5

This is absolutely necessary for each installation. If you have more than one IMLAC and are not developing new software very fast, one console to share between them should be enough. 5a

Tape cassette: 6

A good buy? They are cheap and a very handy way to store programs. Fast paper tape hardware is a reasonable substitute except for much higher price and the bulkiness of paper tape. 6a

Horizontal tube: 7

If you get the long vector hardware and must have long lines of text, you can get this option. It simply provides more X points and squashes the Y points together slightly. 7a

"Long vector" hardware: 8

If you are going to display anything but characters you must have this option. Its main effect is to change the coordinate

system on the screen from a rather peculiar 720-point grid to a well-behaved 1024-point grid. 8a

It is true that the number of characters per line in the smallest comfortable size goes down from 80 to 70. This is because the character sizes are a little smaller and size 1 (the normal size without l.v.h.) is too small to read easily. 8b

This option does not display long vectors any faster, they just take a lot less core space (3 words per vector rather than 1/2 word per step of 3 or fewer points). Speed is still limited to 3 points in each coordinate every 2 microseconds, or about 700 microseconds for a full-screen line. 8c

8-level display subroutine stack: 9

Useful for graphics: since the vectors are all relative, this allows you to store picture prototypes as subroutines. IMLAC/NLS doesn't use it because the storage allocation scheme requires backpointers anyway and the display list has the form of a tree. 9a

Internal interrupts: 10

You can get an interrupt when the standard 25 ms clock goes off or when the display reaches a halt instruction. This is very worthwhile, especially if your display list is composed of several pieces linked only by software. 10a

External interrupts: 11

If you are running a serial interface above 1800 baud (and even this is pushing it), the IMLAC will probably not be able to keep up with it unless you build some extra buffering or get the option which gives an interrupt on every character. The single character of standard buffering only gives you one BIT time to get the character out. 11a

If you get the light pen you must get this option. For other graphical input devices, the internal interrupts should be enough if a 25 ms sampling rate is adequate since you can get an interrupt from the clock. 11b

Light pen: 12

My experience has been that a mouse is the best device for pointng and that a tablet is the best for drawing. The IMLAC light pen can also be quite expensive since it requires the external interrupt facility. 12a

Mouse and keyset: 13

These are available from Cybernex for about $800 installed. They require no other options. They are very valuable for running NLS and also can substitute for the light pen + buttons found on the IBM 2250. 13a

<JOURNAL>7493.NLS;1, 13-AUG-71 16:27 CHI ; Title: Author(s): L. Peter Deutsch/LPD; Distribution: Charles H. Irby, John T. Melvin, Albert Vezza, W. Jack Bouknight, Ira W. Cotton, Steve D. Crocker, Karl C. Kelley, J. C. R. Licklider, John W. McConnell, Robert M. Metcalfe, Edwin W. Meyer, James C. Michener, James A. Moorer, Jon B. Postel, Ken Pogran, Ron M. Stoughton/CHI JTM AXV WJB IWC SDC KCK JCRL JWM RMM EWM JCM JAM JBP KP RMS; Keywords: imlac configuration guide options;
Sub-Collections: ARC NIC ; Clerk: CHI;
Origin: <DEUTSCH>IMCONF.NLS;6, 12-AUG-71 15:18 CHI ;