CONTENTS OF NLS USER'S GUIDE
(page nos. are in parentheses)

Centerdot, pointers, branchonly, tree display.

Note on On-Line Version of this Document

This file also exists in an on-line version with some on-line aids built into it. The on-line version may be found in KDF as (Casseres)NLIST.

NLIST contains links to (Casseres)CONAN, (Casseres)LINKS, and (Casseres)INFOR, which describe the content analyzer, the link system, and the information-retrieval system respectively. These files are also given as hard copy in the User's Guide.

The null statement of NLIST contains some useful names to jump to.

NLIST is designed to be viewed with statement numbers off. The branch-only feature may also be useful.

A number of trails are built into NLIST; they may be entered from Statement (trailset).

This hard-copy version is in KDF as (Casseres)NLUG.

Keyset and VIEWSPEC Codes

(keysetex) The keyset has four cases called 0, 1, 2, and 3. The center and left-hand buttons on the mouse are used for specifying case as follows: Case 0, neither button; Case 1, center button; Case 2, left button; Case 3, both buttons. The buttons are held down while striking chords on the keyset.

Case 0 contains lower-case letters, comma, period, semicolon, question-mark, and space.

Case 1 contains upper-case letters, less-than, greater-than, colon, backslash, and tab.

Case 2 contains various punctuation characters for chords 1-15, digits from 0 to 9 for chords 16-25, more punctuation for chords 26-29, and ALTMODE and carriage-return for chords 30 and 31.

Case 3 contains VIEWSPECs and centerdot.

Codes for Keyset and VIEWSPECs

NOTE: Details on the effects of the various VIEWSPECs are given in the document VIEWS.

Keyset Codes

00001	a	A	!	L=L-1
00010	b	B	"	L=L+1
00011	c	C	#	L=ALL
00100	d	D	$	L=1
00101	e	E	%	L relative
00110	f	F	&	recreate display
00111	g	G	'	branch-only on
01000	h	H	(	branch-only off
01001	i	I	)	content-analyzer on
01010	j	J	@	content-analyzer off
01011	k	K	+	trail on
01100	l	L	-	trail off
01101	m	M	*	statement numbers on
01110	n	N	/	statement numbers off
01111	o	O	0	frozen statements on

10001	p	P	0	frozen statements off
10001	q	Q	1	T=T-1
10010	r	R	2	T=T+1
10011	s	S	3	T=ALL
10100	t	T	4	T=1
10101	u	U	5	pointers on
10110	v	V	6	pointers off
10111	w	W	7	L=T=ALL
11000	x	X	8	L=T=1
11001	y	Y	9	blank lines on
11010	z	Z	=	blank lines off
11011	;	<	[	(nothing)
11100	_	>	]	(nothing)
11101	:	?	+	(nothing)
11110	?	\	ALT	centerdot
11111	SP	TAB	CR	(nothing)

Capital-Letter VIEWSPECs

A -- indenting on

B -- indenting off

C -- names on

D -- names off

E -- clip picture and show (see vectors)

F -- show picture only if it fits (see vectors)

G -- display file as tree structure (see tree)

H -- display file as normal text (see tree)

I --keyword reordering on (see document on information-retrieval system)

J --keyword reordering off (see document on information-retrieval system)

K -- display of statement signatures on

L -- display of statement signatures off

(VIEWSPEC) Control of VIEWSPECs

The VIEWSPECs are used as parameters to control the way in which statements are displayed.

VIEWSPECs may be controlled in three ways: during certain commands such as Jump or Load, with the View Set command, or from the keyset in Case 3. (The viewspecs may also be set from the keyboard with the right-hand and center buttons on the mouse down, i.e. in Case 3 position.)

During the Jump and Load commands (and a few others), there is a point where the VIEWSPECs in the upper left-hand corner of the display become large and are accessible to change. They may then be changed by typing them in from the keyboard or keyset as upper- or lower-case letters.

The View Set command (q) may be used to achieve exactly the same effect.

Case 3 may be used to set all of the VIEWSPECs that are not capital letters, as shown in the table of keyset codes.

This may be done at any time.

After VIEWSPECs have been given in this fashion, it is necessary to hit Chord 00110, Case 3, for "new View," before the new VIEWSPECs will become effective.

Relative Level Control

The code "e" causes the subsequent setting of L to be interpreted relative to the level of the first statement in the new display.

For example, suppose that a Jump is being made to a third-level statement. If "e" is given with no subsequent codes for control of L, the result is L=3. If "eb" is given, the result is L=4; if "e3ba" is given, the result is L=5.

Thus it is possible to get an appropriate setting of L without knowing the appropriate absolute value, simply by specifying a relative value.

Six VIEWSPECs that are not self-evident are displayed as two lines in the upper left-hand corner of the screen.

The top line shows "L" and "T," which appear either as numbers or as the word "all." "L" determines how many levels of statements will be displayed (see level), and "T" determines how many lines of each statement will be displayed.

EXCEPTION: When an "e" is given to set L as a relative value, the area that normally shows L and T shows "R+N" instead. The "n" appears as a number; thus when "e" is given, "R+0" is displayed; if this is followed with a "b," "r+1" is displayed.

The second line shows four VIEWSPECs: g or h for the branch-only parameter on or off (see branchonly); i or j for content-analyzer on or off (see analyzer); k or l for trail feature on or off (see trails); and u or v for pointers displayed or not displayed (see pointer).

Introduction to Command Descriptions

NOTES on Syntax Equations

The capital letters at the beginning of each equation are the control letters entered from keyboard or keyset to specify the command.

CA means "command accept;" this is done by pressing either of the two CA keys on the keyboard, or the right-hand button on the mouse.

Square brackets are used to incicate selections made with the bug. Thus [c] means that a character is selected, [v] means that a visible string is selected, and so forth. If two characters are to be selected, they are shown as [c]1 and [c]2.

To select an entity with the bug, move the mouse until the bug points at any character within the entity, then push CA. An "o" will appear over the character you have pointed to. If this is the character you wanted to point to, push CA again and the selection is completed. If the "o" is over the wrong character, you can press the left-hand mouse button or the "backspace" key and start the selection again.

"LIT" means any string of characters input from the keyboard or keyset.

"NUMBER" means any number entered from the keyboard or keyset.

The slash (/) means "or."

The dollar sign ($) by itself means "any number of" (from 0 to 1000). The construct m$ means "any number equal to or greater than m;" the construct $n means "any number equal to or less than n;" the construct m$n means "any number from m to n."

The term VIEWSPEC in a syntax equation means that VIEWSPECs may be set.

For example, the syntax equation for the Replace Text command (see RT) is R T [c]1 [c]2 LIT CA. This means "type r; type t; select a character; select a second character; type a string of characters; do a command accept."

Delete Commands

(dc) Delete Character

Syntax: D C [c] CA

Semantics: The selected character is deleted.

(dw) Delete Word

Syntax: D W [w] CA

Semantics: The selected word is deleted.

(dv) Delete Visible

Syntax: D V [v] CA

Semantics: The selected visible string is deleted.

(di) Delete Invisible

Syntax: D I [i] CA

Semantics: The selected invisible string is deleted.

(dt) Delete Text

Syntax: D T [c]1 [c]2 CA

Semantics: The selected text string is deleted from c1 to c2.

(ds) Delete Statement

Syntax: D S [s] CA

Semantics: The selected statement is deleted.

(db) Delete Branch

Syntax: D B [s] CA

Semantics: The selected branch (Statement s and all of its substructure) is deleted.

(dp) Delete Plex

Syntax: D P [s] CA

Semantics: The selected plex (the branch headed by Statement s plus all other branches which are in the same sublist) is deleted.

(dg) Delete Group

Syntax: D G [s]1 [s]2 CA

Semantics: Statements s1 and s2 must be in the same sublist. The selected group (the branches headed by Statements s1 and s2, and all intervening branches) is deleted.

(dd) Delete Drawing

Syntax: D D [s]1 CA

Semantics: The drawing associated with the selected statement is deleted.

Insert Commands

(ic) Insert Character

Syntax: I C [c] LIT CA

Semantics: LIT is inserted immediately after the selected character.

(iw) Insert Word

Syntax: I W [w] LIT CA

Semantics: LIT is inserted after the selected word, with a SPACE between.

(iv) Insert Visible

Syntax: I V [v] LIT CA

Semantics: LIT is inserted after the selected visible, with a SPACE between.

(it) Insert Text

Syntax: I T [c] LIT CA

Semantics: LIT is inserted after the selected character, with a SPACE between. The action is identical to (ic).

(ii) Insert Invisible

Syntax: I I [I] LIT CA

Semantics: LIT is inserted immediately after the selected invisible string.

(is) Insert Statement

Syntax: I S [s] LEVADJ SPACE LIT CA

Semantics: LIT becomes the text of a new statement (or set of statements—see centerdot), following the selected statement.

(ib) Insert Branch

Syntax: I B [s] LEVADJ SPACE LIT CA

Semantics: LIT becomes the text of a new statement (or set of statements--see centerdot), following the branch headed by the selected statement.

(ip) Insert Plex

Syntax: I P [s] LEVADJ SPACE LIT CA

Semantics: LIT becomes the text of a new statement (or set of statements--see centerdot), following the selected plex.

(ig) Insert Group

Syntax: I G [s] LEVADJ SPACE LIT CA

Semantics: LIT becomes the text of a new statement (or set of statements--see centerdot), following the selected statement. The effect is identical to (is).

(iq) Insert QED Branch

Syntax: I Q [s] (Y/(not Y)/CA) FILENAME CA

Semantics: Note that no LEVADJ is possible.

In response to the feedback message CONVERT CASE, a Y or CA means YES; any other character means NO.

If YES, all characters will be read as lower-case except those immediately preceded by a backslash: these will be upper-case and the backslashes will disappear.

If NO, all letters will be read by NLS as upper-case letters; backslashes will be taken literally.

The text of the QED file becomes a branch following the selected branch, at the same level. Normally, if the QED file was not created from NLS with the odq command, there will be some garbage in the new branch.

Replace Commands

(rc) Replace Character

Syntax: R C [c] LIT CA

Semantics: The selected character is replaced by LIT.

(rw) Replace Word

Syntax: R W [w] LIT CA

Semantics: The selected word is replaced by LIT.

(rv) Replace Visible

Syntax: R V [v] LIT CA

Semantics: The selected visible string is replaced by LIT.

(rt) Replace Text

Syntax: R T [c]1 [c]2 LIT CA

Semantics: The selected text is replaced by LIT.

(ri) Replace Invisible

Syntax: R I [i] LIT CA

Semantics: The selected invisible string is replaced by LIT.

(rs) Replace Statement

Syntax: R S [s] LIT CA

Semantics: The text of the selected statement is replaced by LIT (see centerdot).

(rb) Replace Branch

Syntax: R B [s] LIT CA

Semantics: The branch defined by the selected statement is replaced by LIT. (see centerdot).

(rp) Replace Plex

Syntax: R P [s] LIT CA

Semantics: The plex defined by the selected statement is replaced by LIT (see centerdot).

(rg) Replace Group

Syntax: R G [s]1 [s]2 LIT CA

Semantics: The group defined by the selected statements is replaced by LIT. (see centerdot).

Move Commands

(mc) Move Character

Syntax: M C [c]1 [c]2 CA

Semantics: Character c2 is moved so that it appears immediately after Character c1.

(mw) Move Word

Syntax: M W [w]1 [w]2 CA

Semantics: Word w2 is moved so that it appears immediately after Word w1, with spaces and punctuation as they should be.

(mv) Move Visible

Syntax: M V [v]1 [v]2 CA

Semantics: Visible v2 is moved so that it appears immediately after Visible v1, with spaces and punctuation as they should be.

(mt) Move Text

Syntax: M T [c]1 [c]2 [c]3 CA

Semantics: Characters c2 and c3 are the first and last characters of a text string; the string is moved so that it immediately follows c3.

(mi) Move Invisible

Syntax: M I [i]1 [i]2 CA

Semantics: The string i2 is moved to follow i1.

(ms) Move Statement

Syntax: M S [s]1 [s]2 LEVADJ CA

Semantics: Statement s2 is moved so as to follow Statement s1, at a level determined by the LEVADJ.

(mb) Move Branch

Syntax: M B [s]1 [s]2 LEVADJ CA

Semantics: The branch headed by Statement s2 is moved so as to follow the branch headed by Statement s1, at a level determined by the LEVADJ.

(mp) Movie Plex

Syntax: M P [s]1 [s]2 LEVADJ CA

Semantics: The plex defined by Statement s2 is moved so as to follow Statement s1, at a level determined by the LEVADJ.

(mg) Movie Group

Syntax: M G [s]1 [s]2 [s]3 LEVADJ CA

Semantics: The group of branches defined by Statements s2 and s3 is moved so as to follow Statement s1, at a level determined by the LEVADJ.

(md) Movie Drawing

Syntax: M D [s]1 [s]2 CA

Semantics: The drawing associated with s2 is moved to s1.

Copy Commands

(cc) Copy Character

Syntax: C C [c]1 [c]2 CA

Semantics: Character c2 is copied immediately after Character c1.

(cw) Copy Word

Syntax: C W [w]1 [w]2 CA

Semantics: Word w2 is copied immediately after Word w1.

(cv) Copy Visible

Syntax: C V [v]1 [v]2 CA

Semantics: Visible v2 is copied immediately after Visible v1.

(ci) Copy Invisible

Syntax: C I [i]1 [i]2 CA

Semantics: Invisible v2 is copied immediately after Invisible v1.

(ct) Copy Text

Syntax: C T [c]1 [c]2 [c]3 CA

Semantics: Text from Character c2 to Character c3 is copied immediately after Character c1.

(cs) Copy Statement

Syntax: C S [s]1 [s]2 LEVADJ CA

Semantics: Statement s2 is copied immediately after Statement s1, at a level determined by the LEVADJ.

(cb) Copy Branch

Syntax: C B [s]1 [s]2 LEVADJ CA

Semantics: The branch determined by Statement s2 is copied immediately after Statement s1, at a level determined by the LEVADJ.

(cp) Copy Plex

Syntax: C P [s]1 [s]2 LEVADJ CA

Semantics: The plex determined by Statement s2 is copied immediately after Statement s1, at a level determined by the LEVADJ.

(cg) Copy Group

Syntax: C G [s]1 [s]2 [s]3 LEVADJ CA

Semantics: The group determined by Statements s2 and s3 is copied immediately after Statement s1, at a level determined by the LEVADJ.

(cd) Copy Drawing !

Syntax: C D [s]1 [s]2 CA

Semantics: The drawing associated with s2 is copied to s1.

Break and Join Commands

(bs) Break Statement

Syntax: B S [v](LEVADJ CA

Semantics: The statement is broken after the selected visible. The LEVADJ adjusts the level of the new statement made up of the last part of the original statement.

(bj) Join Statement

Syntax: B J [s]1 [s]2 CA

Semantics: The text of s2 is appended to s1, and s2 is deleted. If s2 has a sublist, the sublist is moved to follow s1.

Jump Commands

(jo) Jump to Origin

Syntax: J O VIEWSPEC CA

Semantics: The display start is positioned to the first statement.

(ji) Jump to Identity

Syntax: J (I/null) [s] VIEWSPEC CA

Semantics: The display start is positioned to the selected statement. Note that the I in the command specification may be omitted.

(ju) Jump to Up

Syntax: J U [s] VIEWSPEC CA

Semantics: The display start is positioned to the statement of which the selected statement is a substatement.

(jd) Jump to Down

Syntax: J D [s] VIEWSPEC CA

Semantics: The display start is positioned to the first substatement of the selected statement.

(js) Jump to Successor

Syntax: J S [s] VIEWSPEC CA

Semantics: The display start is positioned to the list successor of the selected statement.

(jp) Jump to Predecessor

Syntax: J P [s] VIEWSPEC CA

Semantics: The display start is positioned to the list predecessor of the selected statement.

(jh) Jump to Head

Syntax: J H [s] VIEWSPEC CA

Semantics: The display start is positioned to the first statement in the list where the selected statement is found.

(jt) Jump to Tail

Syntax: J T [s] VIEWSPEC CA

Semantics: The display start is positioned to the last statement in the list where the selected statement is found.

(je) Jump to End

This command expects a third letter. In each case, the command is similar to one of the other Jump commands, except that it considers a branch instead of a statement. The last statement of the branch is placed at the top of the display.

(jei) Jump to End of Identity

Syntax: J E I [s] VIEWSPEC CA

Semantics: The selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jen) Jump to End of Name

Syntax: J E N ([w]/SPACE LIT CA) VIEWSPEC CA

Semantics: The selected name determines a statement; the statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jes) Jump to End of Successor

Syntax: J E S [s] VIEWSPEC CA

Semantics: The successor of the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jep) Jump to End of Predecessor

Syntax: J E P [s] VIEWSPEC CA

Semantics: The predecessor of the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jeu) Jump to End of Up

Syntax: J E U [s] VIEWSPEC CA

Semantics: The source of the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jed) Jump to End of Down

Syntax: J E D [s] VIEWSPEC CA

Semantics: The first substatement of the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jeh) Jump to End of Head

Syntax: J E H [s] VIEWSPEC CA

Semantics: The head of the list containing the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jet) Jump to End of Tail

Syntax: J E T [s] VIEWSPEC CA

Semantics: The tail of the list containing the selected statement determines a branch, and the last statement in that branch is placed at the top of the display.

(jeo) Jump to End of Origin

Syntax: J E O VIEWSPEC CA

Semantics: The last statement in the file is placed at the top of the display.

(jev) Jump to End of Vector Label

Syntax: J E V ([v]/SPACE LIT CA) VIEWSPEC CA

Semantics: This is identical to jen (Jump to End of Name) except that a vector label (see label) is selected instead of a word in text. The selected label or the LIT is used as a name and the end of the branch determined by the named statement is placed at the top of the display.

(jb) Jump to Back

Syntax: J B CA VIEWSPEC CA

Semantics: The display start is positioned to the statement immediately preceding the selected statement (i.e., the end of the sublist of the predecessor), if this statement is a "visible" statement, i.e. one which may be displayed under the current VIEWSPECs. If it is not, then the last previous "visible" statement is used for the display start.

(jn) Jump to Name

Syntax: J N ([w]/SPACE LIT CA) VIEWSPEC CA

Semantics: A statement name is specified by either a word-selection or a literal entry from the keyboard or keyset. When the command is executed, the statement with the specified name is placed at the top of the display.

If the specified name does not exist, the command is aborted with the message "no such name." If more than one statement with the specified name exists, the command is aborted with the message "duplicate name."

(jv) Jump to Vector Label

Syntax: J V ([v]/SPACE LIT CA) VIEWSPEC CA

Semantics: A statement name is specified by either a vector-label selection (see label); or a literal entry from the keyboard or keyset. When the command is executed, the statement with the specified name is placed at the top of the display.

If the specified name does not exist, the command is aborted with the message "no such name." If more than one statement with the specified name exists, the command is aborted with the message "duplicate name."

(jl) Jump to Link

NOTE: For the correct syntax of a link, see document LINKS.

There are two cases of this command, depending on whether the link refers to a location in the current file or in another file (which must be on the RAD).

Syntax (within file): J L [l] VIEWSPEC CA

Semantics: The statement defined by the link is placed at the top of the display, and the VIEWSPECs given in the link are placed in effect, unless they are changed by manual input during the command. The new view is entered in the next location in the intrafile ring: for details on this see document LINKS.

Syntax (out of file): J L [l] CA CA

Semantics: The statement defined by the link (in the file defined by the link) is placed at the top of the display, and the VIEWSPECs given in the link are placed in effect; they cannot be changed by manual input during the command. The new view is entered in the next location in the interfile stack: for details on this slee document LINKS.

(jr) Jump to Return

Syntax: J R CA

Semantics: Whenever a jump within the current file is executed (except for this one and ja), the system keeps track of it. A "ring" is maintained, containing all the views that have been used in sequence. This command causes a return to the previous view; no change is made in the ring itself. For details on this see document LINKS.

NOTE: Because of a bug, the Jump to End commands will cause erroneous entries in the ring.

(ja) Jump to Ahead

Syntax: J A CA

Semantics: Whenever a jump within the current file is executed (except for this one and jr), the system keeps track of it. A "ring" is maintained, containing all the views that have been used in sequence. This command (which can only be used meaningfully after jr has been used at least once) causes a move "forward" along the ring; no change is made in the ring itself. For details on this see document LINKS.

NOTE: Because of a bug, the Jump to End commands will cause erroneous entries in the ring.

(jf) Jump to File

Expects a third letter to specify jfl, jfr, jfa, jfw, or jfc.

(jfl) Jump to File Link

NOTE: For the correct syntax of a link, see document LINKS.

Syntax: (intrafile link) J F L [link] CA

Syntax: (interfile link) J F L [link] CA CA

Semantics: The operation of this command is the same as Jump to Link.

(jfr) Jump to File Return

Syntax: J F R [(CA/character) CA

Semantics: Whenever a file is loaded with Load File or a jump from one file to another is executed (except under this command and jfa), the system keeps track of it. A stack is maintained, containing all the views that have been used in sequence. This command causes a move backward along the stack, to the file previously viewed. No new entry is made in the stack. For details on this see document LINKS.

When the characters J F R have been typed, the system will display the name of the file to be jumped to. If any character is typed instead of the CA, the system will go back one more step on the stack and display another filename.

NOTE: Unlike the rings maintained by the jf and ja commands within a file, this stack will not work as a ring. As a rule, a move past either end of the stack will cause the message ILLEGAL ENTITY and the command will be aborted.

(jfa) Jump to File Ahead

Syntax: J F A (CA/character) CA

Semantics: Whenever a file is loaded with Load File or a jump from one file to another is executed (except under this command and jfr), the system keeps track of it. A stack is maintained, containing all the views that have been used in sequence. This command (which cannot be used meaningfully unless jfr has been used at least once) causes a move forward along the stack. No new entry is made in the stack. For details on this see document LINKS.

When the characters J F A have been typed, the system will display the name of the file to be jumped to. If any character is typed instead of the CA, the system will go forward one more step on the stack and display another filename.

NOTE: Unlike the rings maintained by the jf and ja commands within a file, this stack will usually not work as a ring. As a rule, a move past either end of the stack will cause the message ILLEGAL ENTITY and the command will be aborted.

(jfw) Jump to File Working Copy

Syntax: J F W CA

Semantics: The WORKING COPY file is opened and displayed. The view will be the same as the last view of the WORKING COPY file. No new entry is made in the stack.

NOTE: Please see the discussion of the WORKING COPY file, in the document LINKS.

(JFC) Jump to File Current

Syntax: J F C CA

Semantics: The "current" file is the one currently indicated by the pointer in the interfile stack. Usually the pointer indicates the file being displayed and this command is meaningless. However, when the working copy file is being displayed, this command is the means to return to the file previously displayed. No new entry is made in the stack.

NOTE: Please see the discussion of the WORKING COPY file, in the document LINKS.

Pointer Commands

(pf) Pointer Fix

Syntax: P F [c] [$3(CHAR)] CA

Semantics: The pointer name (up to 3 characters) is attached to the specified character.

(pl) Pointer List Show

Syntax: P L

Semantics: As soon as the command is specified, the display is replaced by a list of all the pointers that have been defined. To return to the display, enter a CA.

(pr) Pointer Release. Expects a third letter to specify an entity. The options are as follows:

(pra)] Pointer Release All

Syntax: P R A CA

Semantics: All pointers are deleted throughout the text.

(prs)] Pointer Release Statement

Syntax: P R S [s] CA

Semantics: All pointers in the selected statement are deleted.

(prw): Pointer Release Word

Syntax: P R W [w] CA

Semantics: All pointers in the selected word are deleted.

(prt): Pointer Release Text

Syntax: P R T [c]] [c]2 CA

Semantics: All pointers in the selected text are deleted.

(q) View Set Commands

Syntax: Q (a/ ... /z/A/ ... /L) CA

Semantics: The command makes the VIEWSPECs accessible for change - just as they are during a Jump command, for example. The lower-case letters have the effect that they have from the keyset. The capital letters have the following effects:

A -- indenting on

B -- indenting off

C -- names on

D -- names off

E -- clip picture and show (see vectors)

F -- show picture only if it fits (see vectors)

G -- display file as tree structure (see tree)

H -- display file as normal text (see tree)

I --keyword reordering on (see document on information-retrieval system)

J --keyword reordering off (see document on information-retrieval system)

K -- display of statement signatures on

L -- display of statement signatures off.

Set Commands

(sc) Set Character

Syntax: S C (C/L/I/R/B/N/F/S/U/W) CA [c] CA

Semantics: The selected character becomes upper-case, lower-case, etc. according to the following codes:

C -- capital

L -- lower-case

I -- italic

R -- roman

B -- boldface

N -- no boldface

F -- flickering

S -- solid (nonflickering)

U -- underline

W -- no underline

(sw) Set Word

Syntax: S W (C/L/I/R/B/N/F/S/U/W) CA [w] CA

Semantics: The selected word becomes upper-case, lower-case, etc. according to the codes given in sc.

(sv) Set Visible

Syntax: S V (C/L/I/R/B/N/F/S/U/W) CA [v] CA

Semantics: The selected visible becomes upper-case, lower-case, etc. according to the codes given in sc.

(si) Set Invisible

Syntax: S I (C/L/I/R/B/N/F/S/U/W) CA [i] CA

Semantics: The selected invisible becomes upper-case, lower-case, etc. according to the codes given in sc.

(ss) Set Statement

Syntax: S S (C/L/I/R/B/N/F/S/U/W) CA [s] CA

Semantics: The selected statement becomes upper-case, lower-case, etc. according to the codes given in sc.

(st) Set Text

Syntax: S T (C/L/I/R/B/N/F/S/U/W) CA [t]1 [t]2 CA

Semantics: The selected text becomes upper-case, lower-case, etc. according to the codes given in sc.

Freeze Commands

(fs) Freeze Statement

Syntax: F S [s] VIEWSPEC CA

Semantics: The selected statement is frozen, with the specified view. It will appear at the top of the screen whenever frozen statements are being shown.

(fr) Release Statement

Syntax: F R [s] CA

Semantics: The selected statement is unfrozen. The selection may be in the frozen area of the display or in the normal viewing area.

(fa) Release All

Syntax: F A CA !

Semantics: All frozen statements are unfrozen.

Alarm Commands

(ac) Alarm Clock Set

Syntax: A C NUM1 CA NUM2 CA NUM3 CA

Semantics: NUM1, NUM2, and NUM3 are the hour, minute, and second to which the alarm clock is set.

(at) Alarm Timer Set. Same as (ac) but relative to present time.

Load Commands

(l) Load Checkpoint

Syntax: L CA VIEWSPEC CA

Semantics: The checkpoint file is loaded into NLS and displayed, beginning at the first statement.

(lf) Load File

Syntax: L F [filename] CA VIEWSPEC CA

Semantics: The file specified by the filename is loaded into NLS and displayed, beginning at the first statement.

A new entry is made in the interfile stack, which is used by the Jump to File Return, Jump to File Ahead, and Jump to File Current commands. For details, see document LINKS.

Output Commands

(o) Output Checkpoint

Syntax: O CA

Semantics: The working text is written out on the "checkpoint" file, which is automatically created if necessary.

(of) Output File

Syntax: O F (LIT/null) CA CA

Semantics: The LIT specifies a filename, which must be a RAD file; if it is an existing RAD file, it must be sequential. The working text is written out on the specified file.

The system will offer the name of the last file the working copy has been written on, or if it has not been written out, the name of the file it was loaded from. If this is the desired filename, the LIT may be omitted.

(od) Output Device. Expects a third letter to specify the "device." The options are as follows:

(odq) Output Qed File

Syntax: O D Q (LIT/[filename]) CA

Semantics: The LIT or filename specifies a file. The working text is output through Pass 4 in QED format. Note that lower-case letters will come through as garbage, and that directives will be executed.

Pass 4 is automatically initialized with the following directives: psw=0, ind=0, rtj=0, and hsw=0.

The origin should not be on the display when ODQ is executed.

(odp) Output Printer File

Syntax: O D P (LIT/[filename]) CA

Semantics: The LIT or filename specifies a file. The working text is output through Pass 4 in printer format.

This file may then be output to the printer itself by using the program PASS 4 KLUDGE PRINT from Teletype 1 in the control room.

(odd) Output Dura File

Syntax: O D (LIT/[FILENAME])

Semantics: The FILENAME "8-level" causes output on punched paper tape.

(odf) Output Film File

Syntax: O D F (LIT/[FILENAME])

Semantics: A file is written in the correct format for CRT-to-film processing.

(om) Output MOL

Syntax: O M 2$2:(Y/CA OR NOT Y/CA) LIT/[filename] CA

Semantics: MOL source code from the displayed file is output to the MOL compiler, which in turn writes object (assembly) code on a designated file.

Two optional choices are offered with this command.

When the M is typed, the word "reentrant" appears in the command feedback Line. If reentrant code is desired, the word is accepted with a CA or a Y (for "yes"). Any other character causes the word "reentrant" to be rejected and disappear.

Next the word "temps" appears, referring to generation of temporary storage. If temporary storage is to be generated, the word is accepted with Y or CA; otherwise it is rejected with any other character.

Finally a file name is requested and entered either as a literal or by a bug selection.

(os) Output SPL

Syntax: O S LIT/[filename] CA

Semantics: SPL source code from the displayed file is output to the SPL compiler, which in turn writes object (assembly) code on the designated file.

(ot) Output Meta

Syntax: O T LIT/[filename] CA

Semantics: Tree Meta source code from the displayed file is output to the Tree Meta compiler, which in turn writes object (assembly) code on the designated file.

Execute Commands

(ec) Execute Content Analyzer

For a discussion of the content analyzer, see the document on the analyzer. This command is also used for the trail feature: see trails.

Syntax: E C [p] CA

Semantics: P is a pattern which specifies a content requirement for statements to be displayed. The pattern is compiled to produce a content-analyzer program which will cause only the statements meeting the requirement to appear on the display.

NOTE: This command does not put the content analyzer into action; it merely causes a pattern to be compiled. If the pattern has been incorrectly specified, the message "syntax error" will appear; otherwise, the message "successful compilation" will appear.

To put the content analyzer into effect, use the VIEWSPEC "i"; to turn it off use the VIEWSPEC "j". See VIEWSPEC.

(eo) Execute OOPS

Syntax: E O CA !

Semantics: This command is for experimental use. It is a way of deliberately making NLS blow up on you.

Whenever an OOPS occurs (naturally or on command), a system file is automatically written which contains useful information for system programmers.

(ef) Execute File Cleanup

Syntax: E F CA!

Semantics: This will clean out certain invisible problems in a file and produce a display of messages showing what kinds of errors have been found.

IMPORTANT NOTE: File Cleanup does not always correct all the errors it finds.

(es) Execute Status

Syntax: E S CA!

Semantics: As soon as the S is typed, the display shows various items of information about the file. When the CA is hit, the normal display is restored.

(ev) Execute Viewchange

NOTE: The syntax and semantics are given in a separate section because of their complexity.

Vector Commands

(v) Vector Package:

Syntax: V [Is] CA ;

Semantics: The specified statement is placed at the top of the screen and the rest of the screen is cleared. The commands in the vector package may then be used to create or modify a picture attached to the statement. See vectors.

Keyword Commands

(ks) Keyword Select

Syntax: K S [s] CA

Semantics: When the statement is selected, the name is displayed to the right of the command feedback line. When the CA is hit, the name becomes a keyword for use in the keyword system (see keywords).

(kw) Keyword Weight

Syntax: K W [s] $(NUMBER) CA

Semantics: When the statement is selected, the weight of its name (as a keyword) is displayed to the right of the command feedback line. When numbers are typed in, they change this weight, and when the CA is hit the number displayed becomes the new weight of the keyword (see keywords). If the CA is hit without typing any digits, the weight is unchanged.

(kf) Keyword Forget

Syntax: K F [s] CA

Semantics: When the statement is selected, the weight of its name (as a keyword) is set to zero (see keywords).

(ka) Keyword Forget All

Syntax: K A CA

Semantics: The weights of all keywords are set to zero (see keywords).

(ke) Keyword Execute

Syntax: K E CA

Semantics: The keyword system is placed in operation to produce an ordered list of referents (see document on information-retrieval system).

(mousebut) Mouse Buttons

Right-Hand Button :

When pushed and released without any intervening input, this button gives a ICA (command accept).

When it is held down while a LIT is entered from keyboard or keyset, this button causes the LIT to be interpreted as a reference to a pointer.

Center Button

When pushed and released without any intervening input, this button gives a ICD (command delete).

When it is held down while a LIT is entered from keyset, this button causes the LIT to be interpreted as Case 1 input.

Left-Hand Button

When pushed and released without any intervening input, this button gives a ibackspace, causing the last input character to be thrown away.

When it is held down while a LIT is entered from keyset, this button causes the LIT to be interpreted as Case 2 input.

Left-Hand and Center Buttons Together :

When pushed and released without any intervening input, this combination gives a backspace-word, causing the last input word to be thrown away.

When it is held down while a LIT is entered from keyset, this combination causes the LIT to be interpreted as Case 3 input.

(vectors) The Vector Package

The vector package allows the user to create simple line drawings with labels, as a part of his file. See v for how to enter the vector package in association with a particular statement.

There will eventually be a means to output the pictures, but for now they are only visible within NLS.

The following commands are valid within the vector package:

Insert Commands

(i v) Insert Vector

Syntax: I V (CA / B / CD) CA

The bug mark is used to determine the endpoints of lines.

Each CA after the first determines a line.

Thus 4 CA's produce 3 lines, with line 1 meeting line 2 at the position of the second bug mark, and line 2 meeting line 3 at the third bug mark.

To "lift your pencil" and break the continuity of the lines type a "B" or a CD.

(i l) Insert Label

Syntax: I L SPACE LIT CA CA

Semantics: After typing the label, hit a CA to attach the label to the bug. The next CA fixes the label in its current position on the screen (rounded off to the nearest position that can be output on the printer).

Move Commands

(m v) Move Vector

Syntax: M V (NUMBER / nothing) CA (A / B) CA

Semantics: The vectors are numbered for identification.

After a vector has been specified by typing its number or accepting the number offered in the name register, hit a CA.

This will cause the ends of the vector to be marked "A" and "B".

Type the letter of the end you wish to move. That letter will be replaced by an "X" and the other letter will go away.

Then position the bug where you want the end to go and hit a CA.

(ml) Move Label

Syntax: M L [l] CA CA

Semantics: When the first CA is hit, the label [l] is attached to the bug and moves with it. The next CA fixes the label in the new position.

(md) Move Drawing

Syntax: M D [point]1 [point]2 CA

Semantics: The two points selected define a translation vector; the entire drawing is moved according to this vector.

Delete Commands:

(dv) Delete Vector

Syntax: D V (NUMBER / nothing) CA

Semantics: Select the vector to be deleted, using its number, and hit a CA.

(dl) Delete Label

Syntax: D L [l] CA

Semantics: The label [l] is deleted.

(dd) Delete Drawing

Syntax: D D CA

Semantics: All vectors and all labels in the drawing are deleted. The command is used for starting over from scratch.

(t) Translate Vector

Syntax: T (NUMBER / nothing) CA (A / B) CA

Semantics: This command is identical to mv in terms of actions by the user to specify which vector and which end.

The vector keeps its original length and angle, but is moved to a new position on the screen.

The specified end is moved to the position of the bug when the CA is hit.

(vv) Vertical

Syntax: V (NUMBER / nothing) CA (A / B) CA

Semantics: The specified end is moved when the CA is hit so that the vector is vertical.

(h) Horizontal

Syntax: H (NUMBER / nothing) CA (A / B) CA

Semantics: The specified end is moved when the CA is hit so that the vector is horizontal.

(g) Grid

Syntax: G CA

Semantics: The grid provides the user a means to draw "pretty pictures".

All positions are rounded off to the points on the grid.

The grid also places lines going through grid points such that they can be output on the printer and still look like straight lines.

The grid is either on or off; after typing "g" to get "grid" in the command feedback line, a CA causes the grid to change state.

Spacing

(sf) Spacing Off

Syntax: S F CA

Semantics: This will set a flag that goes along with the picture telling the display creation routines not to space the statements to leave room for this picture.

(sn) Spacing On

Syntax: S N CA

: Semantics: This is the complementary command to spacing off.
: Since the flag is set for spacing on as the default option,
: this command is necessary only to change the flag back.

(a) Abort

Syntax: A CAI

Semantics: Everything that has been done in the current instance of the vector package is thrown away, the command "Vector Package" is aborted, and it is as if the command had not been given.

(f) Finished

Syntax: F CAI

Semantics: This command returns control from the vector package to NLS proper.

(trails) The Trail Feature

The trail feature is used to set up statements in such a way that only a particular set of statements will be displayed, and in a particular order.

To understand this feature, it is necessary to know roughly what the content analyzer is about; see the document on the content analyzer.

The trail feature works as follows: some pattern is set up as a "trail marker" for a particular set of statements. This pattern is to be compiled with the Execute Content Analyzer command (see ec).

The trail marker is used to mark "turning points" from the normal sequence of statements. Each time the marker appears in a statement, it is followed by a statement name in parentheses. This combination of marker and name is used by the display-creation routines as a signpost to the next statement to be displayed. Between signposts, statements are displayed in normal sequence -- subject to any other VIEWSPECs that may be in effect (see VIEWSPEC).

The Viewchange System

Discussion: This command has an extensive package of subcommands for changing various parameters applying to the display of various entities on the screen. (The syntax description is given further on.)

These entities are the bug (its armed cursor, its active cursor, and the mark it leaves), the command feedback line, the date/time register, the echo register, the name register, the VIEWSPEC feedback area, the working text area, and the tab stops.

The first level of subcommands includes specification of these entities plus the subcommands "save" and "restore."

"Save" is used to store all of the current parameters in a buffer with a number from 1 to 8, and "restore" takes one of these buffers and puts the parameters stored in it into effect.

The second level of subcommands includes the following:

"Character," which applies only to the bug entities and is used to specify the character to be used for plotting each of them.

"Abbreviated," which applies only to the command feedback line and causes it to show only the first letter of each word that it would normally contain.

A special set applying only to the text area:

"Columns" (number of characters in a full line of text)

"Indenting" (number of spaces to indent for each level)

"Lines" (number of lines in text area — currently limited to 20)

"Rows" (same as "lines")

"Vertical increment" (spacing between lines)

"Parameter set," which applies to all entities except tab stops. This subsubcommand has a number of subsubsubcommands.

Syntax: The syntax is given below in hierarchical form, with the input letters at the beginning of each statement. "Parameter Set" is given in a separate branch following the hierarchy.

Execute Viewchange: E V B/C/D/E/N/R/S/T/V

Bug: B D/A/M

Disarmed Cursor: D C/P

Character: C (any character) CA

Parameter Set: P (see below)

Armed Cursor: A C/P

Character: C (any character) CA

Parameter Set: P (see below)

Mark: M C/P

Character: C (any character) CA

Parameter Set: P (see below)

Command Feedback Line: C P/A

Parameter Set: P (see below)

Abbreviated: A CA

Semantics: Only the first letter of each word in the command feedback is displayed.

Date/Time Register: D P

Parameter Set: P (see below)

Echo Register: E P

Parameter Set: P (see below)

Name Register: N P

Parameter Set: P (see below)

Restore: R (digit, 1-8) CA

Semantics: Viewchange conditions previously saved under the digit typed are placed in effect.

Save: S (digit, 1-8) CA

Semantics: The current Viewchange conditions are saved under the digit typed and may be placed in effect later with "Restore."

Tabs: (tab character) 1$(number CA) CA

Semantics: Each number typed (followed by a CA) sets the position of the next tab stop on the display, relative to the previous one (the first one is relative to the margin). An additional CA terminates the command.

Text Area: T C/I/L/P/R/V

Columns: C (number) CA

Semantics: In effect, this sets the number of character positions from the left-hand margin to the right-hand margin. The left-hand margin is fixed.

Indenting: I (number) CA

Semantics: The number typed sets the number of spaces to indent for each level in the statement structure.

Lines: L (number) CA

Semantics: This sets the number of lines available for the text area. It is limited to 20.

Parameter Set: P (see below)

Rows: R (number) CA

Semantics: This sets the number of lines available for the text area. It is limited to 20.

Vertical Increment: V (number) CA

Semantics: This sets the distance between lines in the text area.

VIEWSPEC Area: V !

Parameter Set: P (see below)

Syntax Under "Parameter Set"

Parameter Set: P (B/D/F/H/I/P/S

Boldface On/Off: B $(NOT CA) CA

Semantics: Any character except CA changes "on" to "off" or vice versa.

Display On/Off: D $(NOT CA) CA

Semantics: Any character except CA changes "on" to "off" or vice versa.

Flicker On/Off: F $(NOT CA) CA

Semantics: Any character except CA changes "on" to "off" or vice versa.

Horizontal Increment: H (number) CA

Semantics: This sets the horizontal distance between characters.

Italics On/Off: I $(NOT CA) CA

Semantics: Any character except CA changes "on" to "off" or vice versa.

Position: P (bug selection of point on screen) CA

Semantics: This sets the position of the first character of the entity, unless the entity is "text area." In this case, the horizontal component is ignored and the vertical position of the first line is set.

Size: S (digit 0-3) CA

Semantics: This sets the character size for the entity: 0 is smallest, 1 is normal, and 3 is largest.

Definitions:

(branch) A specified statement, plus all of its substructure -- i.e. all of its substatements, plus all of their substatements, etc.

(bug) The mark on the screen which is moved by the mouse and which is used for selecting (pointing to) entities on the display.

When the bug is "active," i.e. when a selection can be made, it appears as an up-arrow; when it is inactive it appears as a plus sign.

(character) Any letter, digit, punctuation mark, space, tab, or carriage return; an indivisible entity

(chord) A combination of keys on the keyset (see keyset).

For a table showing the meanings of the chords, see keytable.

(file) A complete tree structure of statements with a single root (the origin).

(filename) The name of a file. It appears as the first word in the origin.

(GCHAR) Any space, tab, or carriage return

(group) A subset of a plex, consisting of all branches from one specified branch to another, inclusive.

(head) The first statement in a sublist.

The head is specified by specifying any statement in the sublist.

(invisible) Any consecutive string of gap characters, bounded by (but not including) printing characters or the end of a statement: see printing character, gap character, statement

Specified by selecting any character in the string.

(keyset): The device at the left-hand side of the console. When a combination of keys (a chord) is depressed on the keyset, the effect is the same as striking a key on the keyboard.

For an explanation of the keyset, see keysetex.

(keywords) A content indicator for reference retrieval (see keywords).

(label) A string of text placed in a drawing by means of a command in the vector package. See vectors.

(LEVADJ)| The specification of level when a statement, branch, plex, or group is newly created or moved.

The system will offer a new number of the same level as the preceding number; this can be changed to a higher level by entering a "U" for "UP" or to a lower level by entering a "D" for "down."

(level) The "rank" of a statement (see statement) in the hierarchy of the file (see file).

The level is equal to the number of fields of letters or digits in the statement number; thus Statement 3 is a first-level statement, Statement 4a10g3 is a fifth-level statement, etc. Level is of great importance in understanding the hierarchical structure of an NLS file.

(mouse) The device at the right-hand side of the keyboard. When it is rolled around on the tabletop, it causes the bug to move correspondingly. For an explanation of the three buttons on top of the mouse, see mousebut.

(name) If the first word of a statement is enclosed in parentheses, it is the name of the statement.

The command Jump to Name (see jn) can then be used to place the statement at the top of the display. This is done by entering the name from the keyboard or keyset, or by finding an occurrence of the name as text on the display and pointing to it with the bug.

(origin) The first statement in a file; it contains information about the file, plus any other text the user inserts. It has a level of 0, and hence no statement number.

(pattern) A string of text in a statement which may be compiled via the command Execute Content Analyzer (see ec). When compiled, it produces a program that is used by the content-analyzer feature (see analyzers) or by the trail feature (see trails).

(PCHAR) - Any letter, digit, or punctuation mark

(plex) Another name for a substructure, used in command specifications.

(pointer) A string of up to three characters which is attached to some character in the text with the Pointer Fix command (see pf).

(predecessor) The statement preceding a specified statement in a sublist.

(source) The statement of which a specified statement is a substatement.

(statement) The basic structural unit of a file of text in NLS. Formally, it is a string of text and/or pictures (see vectors) which is bounded at the beginning by the end of the previous statement or the beginning of the file, and bounded at the end by the beginning of another statement or the end of the file.

Statements are arranged in a tree structure or hierarchy and are assigned "statement numbers" which indicate their positions in the structure. Each statement has a number, made up of alternating fields of digits and letters; the number of fields indicates the "level" of the statement (see level).

A statement is specified by selecting any character in the string.

(sublist) The set of all substatements of a specified statement (not including the substatements of the substatements).

(substatement) A statement "X" is called a substatement of another statement "Y" if it is deeper in the structure than "Y," if it follows ("Y," and if there is no intervening higher-order statement. "Y" is called the source of "X." The statement number of "X" will be the same as that of "Y" except that it will have one more field at the end. The value of this field gives its ordinal position in a "sublist" of the substatements of "Y."

A substatement is specified by specifying the source statement.

(substructure) The set of all substatements of a specified statement, plus all their substatements, etc. until no more are found. The set of all branches defined by statements in the sublist of a given statement.

(successor) The statement following a specified statement in a sublist.

(tail) The last statement in a sublist.

The tail is specified by specifying any statement in the sublist.

(text) Any string of characters within a statement, bounded by (and including) two specified characters: see character, statement

(trail) A set of statements in a file, which can be displayed sequentially by using the trail feature. For a discussion of this, see trails.

(vector): A line in a picture. See vectors.

(visible) Any consecutive string of printing characters, bounded by (but not including) gap characters or the end of a statement: see printing character, gap character, statement

Specified by selecting any character in the string. If a single gap character between two visibles is selected, then both visibles are specified.

(word) Any consecutive string of letters and/or digits, bounded by (but not including) any other types of characters or the end of a statement: see statement

Specified by selecting any character in the string. If a single gap character between two words is selected, then both words are specified.

Odds and Ends

(centerdot) The centerdot character may be used in any LIF input to end a statement and start a new statement. LEWADJ is permitted.

(pointers) Use of Pointers

Pointers make it possible to select entities that are not on the display. The entity must have a pointer fixed on it (see pf) for this to be done. To select the entity, hold down the right-hand button on the mouse while entering the name of the pointer from the keyboard or keyset, and then release the button. This is exactly equivalent to making a direct bug selection of the character that has the pointer on it.

(branchonly) The Branch-Only Feature

When the branch-only feature is turned on with the VIEWSPEC g, only one branch at a time is displayed (see branch). In some cases, this gives a less confusing view of the text. It also affects the amount of material output under the Output Device command; if branchonly is on, output ceases as soon as the branch defined by the statement at the top of the screen has been output.

(tree) The Tree-Display Feature

By using the VIEWSPEC capital G, it is possible to see the file as a tree structure instead of text. The tree structure shows the relationships of statements in the file.

All structural commands may be used with the tree display.

To return to the text display, use the VIEWSPEC capital H.

INFOR: KEYWORD INFORMATION-RETRIEVAL SYSTEM

1 The Information-Retrieval System

1A The information-retrieval system permits a user to construct a specially formatted "catalog" file, containing references to other files and capable of being reordered automatically according to some chosen set of weighted keywords. When reordered, the file lists references in order of relevance according to the choice and weighting of keywords.

1B The Catalog File

1B1 The catalog file has two functioning sections: a list of file references pointing to other files, and a list of relevant keywords to be used in retrieving file references.

1B1A Other material may also be included in the file without any effect on the functioning of the retrieval system. For example, since the keyword section is to be studied directly by the user, it may be desirable to group the keyword entries into categories and separate them with headings and subheadings.

1B2 File-Reference Section

1B2A Each file reference is a separate statement beginning with a serial number in parentheses, followed by a link pointing to the referenced file. This is followed by a list of keywords relating to the file, followed by comments on the file.

1B2A1 Only the first item is actually essential to the working of the system, and it need not actually be a serial number; any string of letters and/or digits enclosed in parentheses (i.e. a "statement name" as recognized by NLS) will suffice, as long as it is unique to the particular reference.

1B2A2 The use of serial numbers as "names" in file-reference statements, and the inclusion of the other items, are matters of convenience to the user.

1B3 Keyword Section

1B3A Each keyword must be a single word -- i.e. it must contain no nonprinting characters. Apart from this, it may be any arbitrary string of characters. It is convenient to use short strings of three or four letters standing for longer words or phrases.

1B3B Each entry in the keyword section is a separate statement

with the following format: first the keyword itself, in parentheses, serving as the name of the statement; then the word or phrase for which it stands, plus any comments or other information that may be desired; and finally a special code string (such as an asterisk or a dollar sign followed by a space) followed by a list of serial numbers which are the names of statements in the file-reference section. Each of these serial numbers must be enclosed in parentheses.

1B4 An example is given at the end of this document.

1C Keyword Commands

1C1 This section explains the effects of the keyword commands. Full details on the syntax and control-dialogue procedures may be found in the NLS User Guide.

1C2 The keyword commands operate upon the keywords themselves, i.e. the names of statements in the keyword section of the catalogue. The commands permit the user to select keywords as relevant; assign integer weights to them; change weights; display a list of keywords that have been selected, with their weights; and produce an ordered display of the relevant file references.

1C3 Keyword Select Command

1C3A This command is used to select a given keyword as relevant. It is automatically assigned a weight of 1.

1C4 Keyword Weight Command

1C4A When a keyword is selected under this command, its current weight is displayed (if it has not been previously selected, its weight is zero). The user may then type in an integer which becomes the new weight.

1C5 Keyword List Command

1C5A This command causes display of a list of keywords with nonzero weights.

1C6 Keyword List Weight Command

1C6A This is the same as the "List" command, except that the weights are shown.

1C7 Keyword Forget Command

1C7A When a keyword is selected under this command, its weight

is reset to zero, just as if it had never been selected.

1C8 Keyword Forget All Command

1C8A This command causes all keyword weights to be reset to zero.

1C9 Keyword Execute Command

1C9A This command executes a program which produces an ordered display of statements from the file-reference section of the catalogue.

1C9A1 Each entry for a selected keyword is scanned, and the serial numbers which it contains are noted.

1C9A2 Each of these serial numbers is the name of a statement in the file-reference section: each of these statements is assigned a "score" equal to the weight of the keyword, and this score is accumulated with further references from other keywords.

1C9A3 When all of the selected keywords have been used to score the file references, the file-reference statements with non-zero scores are displayed in order of decreasing score.

1D Example of Catalog File

1D1 Examples -- Keyword Entries

1D1A	(nls) on-line system	*	(u1) (u2) (u3) (u4)
1D1B	(ug) user guides	*	(u1) (u2) (u4)
1D1C	(kse) keyset	*	(u1)
1D1D	(cdp) control-dialog proc.	*	(u1)
1D1E	(anz) content analyzer	*	(u2)
1D1F	(fij) file jumping	*	(u3)
1D1G	(inf) info. retrieval	*	(u4)
1D1H	(vs) view control	*	(u1) (u3)

1D2 Examples -- File-Reference Entries

1D2A (u1) (nlist,1:xnhj) nls,ug,vs,kse,cdp; nls user guide

1D2B (u2) (conan,1:x2bhj) anz,ug,nls; content analyzer user guide

1D2C (u3) (links,1:x2bhj) fij,vs,nls; link jumping and returns

1D2D (u4) (infor,1:x2bhj) inf,ug,nls; information retrieval system

1E Example of Reordering

1E1 Suppose that the system is used on the catalogue shown above. The user has considerable interest in file jumps, so he gives the keyword "fij" a weight of 5. He is also interested in control-dialog procedures, so he gives the keyword "cdp" a weight of 3. Finally, he is also interested in user guides, so he gives "ug" a weight of 1.

1E2 When the command Keyword Execute is given, the following scoring is done:

1E2A The keyword "fij", with a weight of 5, applies to serial number u3; therefore the statement whose name is "u3" is given a score of 5.

1E2B The keyword "cdp", with a weight of 3, applies to serial number u1; therefore the statement whose name is "u1" is given a score of 3.

1E2C The keyword "ug", with a weight of 1, applies to serial numbers u1, u2, and u4; therefore the statements whose names are "u1", "u2", and "u4" are given scores of 1 each. In the case of "u1", this is added to the previous score of 3.

1E3 The final scores are 4 for "u1", 1 for "u2", 5 for "u3", and 1 for "u4". They are then displayed as follows:

(u3) (links,1:x2bhj) fij,vs,nls; link jumping and returns
(u1) (nlist,1:xnhj) nls,ug,vs,kse,cdp; nls user guide
(u2) (conan,1:x2bhj) anz,ug,nls; content analyzer user guide
(u4) (infor,1:x2bhj) inf,ug,nls; information retrieval system

1E3A The user may then access the referenced files by using the Jump to Link command with the links given in the references.

/CONAN, 10/30/68 1529:39 DGC ; ["parenth"];

1 The Content Analyzer

1A The content analyzer feature of NLS permits the user to specify (in a special language) a pattern of content. The analyzer is compiled in real time from the user's specification, and when it is turned on (by a VIEWSPEC parameter) only statements which meet the content specification will appear on the display.

1A1 The pattern specified may be a simple one -- e.g. it may specify a string of characters that must appear somewhere in each statement to be displayed; or it may be complex -- e.g. it may specify a string, to be followed within a given number of words by another specified string, in statements which were created after a certain date by a certain author, and not containing some third specified string.

1A2 The language for specifying content patterns is extremely simple and easy to use for simple cases, but more exacting for complex cases.

1B Pattern-Specification Language

1B1 The Process of Searching a Statement

1B1A When the content analyzer is turned on, each statement in the file is searched, character by character, for the content specified in the pattern. Normally, the search begins with the first character, but it is possible to cause the search to proceed backwards.

1B1B The analyzer uses a pointer to keep track of the search. The pointer always indicates which character is to be examined next, unless something in the pattern causes the pointer to be moved first.

1B1C At any given moment in the search process, the analyzer is searching for one of four types of content entity:

1B1C1 A literal string of characters, such as "abcd" or "13-x" or "ed Mat" or "memory."

1B1C2 A string of character-class variables; these are explained in detail further on. A string of character-class variables might specify "three digits, one after another," or "two letters, followed by any number of spaces, followed by three to five letters or digits."

1B1C3 The date associated with the statement. This is not displayed, but every statement bears the date on which it was created or most recently modified.

1B1C4 The initials associated with the statement. This is not displayed, but every statement bears the initials of the user by whom it was created or most recently modified.

1B1D All of the more complex analysis is achieved by moving the pointer according to the logic of the pattern specification.

1B1D1 For example, if the analyzer is to start at a given point and find either String A or String B, it first looks for string A and then, if String A is not found, the pointer is returned to the starting point and a search is made for String B.

1B2 Basic Elements

1B2A Every pattern ends with a semicolon.

1B2B Every pattern is made up of one or more of the basic entities listed above, combined by operators.

1B2C If the pattern (or some part of it) is to be found anywhere after the point in the statement where the search begins, it is enclosed in square brackets; otherwise it must be the first thing found.

1B2D A string of characters specified as content is enclosed in quotation marks. For convenience, if the string consists of only one character it may be preceded by an apostrophe and the quotation marks omitted.

1B2D1 Examples

1B2D1A ["memory"]; This pattern will cause display of only those statements containing the word "memory" at any point.

1B2D1B "inside"; This pattern will cause only statements beginning with the word "inside" to be displayed.

1B2D1C ['3]; This pattern will cause display of only those statements containing the character "3" at any point.

1B2E Patterns like those shown in the examples above may be strung together; the significance of this is that one item is to be found after the one specified ahead of it.

1B2E1 Examples

1B2E1A ["abc""def"]; This pattern specifies that the string "abc" immediately followed by the string "def"

must appear somewhere in each statement to be displayed. The pattern ["abcdef"]; is exactly equivalent.

1B2E1B ["abc"]["def"]; This pattern specifies that the string "abc" is to be found anywhere in the statement, and anywhere after the "c" the string "def" is to be found.

1B3 Character-Class Variables

1B3A The character-class variables are as follows:

1B3A1 L means any letter

1B3A2 D means any digit

1B3A3 LD means any letter or digit

1B3A4 PT means any printing character (any character except space, tab, and carriage return)

1B3A5 SP means a space

1B3A6 TAB means a tab

1B3A7 CR means a carriage return

1B3A8 NP means any nonprinting character (space, tab, or carriage return)

1B3A9 CH means any character at all.

1B3B Examples

1B3B1 ['.LLL'=D';]; This pattern will cause display of only those statements containing (anywhere) the following content: a period immediately followed by three letters, immediately followed by an equals sign, immediately followed by a digit, immediately followed by a semicolon.

1B3B2 "abcd"SPL D; This will cause display of only those statements beginning with the following content: the string abcd immediately followed by a space, immediately followed by any letter, immediately followed by any digit.

1B3B2A Note that a space is necessary between the L and the D because of a possible ambiguity: The pattern "abcd"SPLD; would mean "the string abcd immediately followed by a space, immediately followed by any letter or digit," because LD means any letter or digit.

1B4 The Dollar Sign (Arbitrary-Number Construct)

1B4A The arbitrary-number construct, in its most general form, is m$n. The meaning is "any number from m to n of occurrences of the following entity."

1B4B When the analyzer has found n occurrences of the specified entity, it also looks ahead to see if there is another occurrence. If there is, the test is considered to have failed. In other words, the limits m and n are absolute.

1B4B1 Example

1B4B1A The pattern 5$11LD; specifies that each staement to be displayed must begin with five to eleven letters and/or digits.

1B4B1A1 A statement beginning with twelve or more letters and/or digits would be rejected by this pattern.

1B4C The m or the n, or both, may be omitted; their assumed values in this case are m=0, n=1000. For all practical purposes, then, the default value of n is "any arbitrary number," since it is very unlikely that any entity will occur 1000 times consecutively.

1B4C1 Examples

1B4C1A The pattern [7$D1$12L$5NP]; specifies that each statement to be displayed must contain the following: seven or more digits immediately followed by one to twelve letters, immediately followed by zero to five nonprinting characters.

1B4C1B The pattern 2$"abc"; specifies that each statement to be displayed must begin with two or more occurrences of the string abc, one after another.

1B5 Grouping by Parentheses

1B5A Parentheses may be used as they are in algebra to group elements. The specifications found within the parentheses are then treated as a single entity for logical purposes.

1B5A1 Example

1B5A1A [3$4(DSPL)1$2NP]; This pattern specifies that each statement to be displayed must contain the following: three or four ocurrences of the string (digit space letter), immediately followed by one or two nonprinting characters.

1B5A1A1 If the parentheses were not used, the 3$4 construct would apply only to the D.

1B5B The square brackets have same grouping the effect as parentheses; however, they are not interchangeable with parentheses because they also mean that the enclosed pattern may be found anywhere after the starting point.

1B6 Operators

1B6A The operators used for combining entities are as follows, in order of decreasing precedence:

1B6A1 - (minus sign): This indicates negation. Thus -LD means a character which is not a letter or a digit.

1B6A1A Example: ["abc"-SP]; This pattern specifies that each statement to be displayed must contain the string abc immediately followed by some character which is not a space.

1B6A2 (space): This indicates concatenation. Thus "abc""xyz"; specifies that the string abc must occur and must be immediately followed by the string xyz.

1B6A2A The space may be omitted unless it is necessary to prevent ambiguity.

1B6A3 / (slash): This indicates alternation. Thus SP/TAB means a character that may be either a space or a tab.

1B6A3A Example: 1$SP/2$3PT; This pattern specifies that each statement to be displayed must begin with either one or more spaces, or two or three printing characters.

1B6A4 NOT: This indicates negation, and is the same as the minus sign except for lower precedence.

1B6A5 AND: This is Logical intersection.

1B6A5A The action of the AND is to return the pointer to the beginning of the search that has just been completed.

1B6A5A1 Example: The pattern ["abc"]AND["xyz"]; causes each statement to be searched first (from the beginning) for the string abc; then, if it is found, the statement is searched again from the beginning for the string xyz. Each statement displayed will contain both strings, but the order in which they occur will be irrelevant.

1B6A5A1A Note that this is different from the pattern ["abc"]["xyz"]; if the AND is not used, the second search is not made from the beginning but from the point just after the end of the first

search. Each statement displayed will then contain both strings, but the string xyz must be somewhere after the string abc. When the AND is used, this restriction will not apply.

1B6A5A1B Note also that the pattern ["abc"AND"xyz"]; is meaningless: it specifies a string that is both "abc" and "xyz".

1B6A6 OR: This is the same as the slash sign, except for the lower precedence.

1B6B Note on Precedence of Operators: As used here, "high precedence" means that when the pattern is parsed, the higher-precedence operators are used first in grouping the elements of the pattern. Thus a high-precedence operator has low "binding power."

1B6B1 Example: Consider the pattern a AND b OR c/-d AND NOT e f; where a, b, c, d, e, and f are pattern elements such as quoted strings or character-class variables.

1B6B2 This is grouped as follows:

1B6B2A The minus sign has the highest precedence, so that we have a AND b OR c/(-d) AND NOT e f;

1B6B2B Next is concatenation, so we have a AND b OR c/(-d) AND NOT (e f);

1B6B2C Next is the slash, so we have a AND b OR (c/(-d)) AND NOT (e f);

1B6B2D Next the NOT, giving a AND b OR (c/(-d)) AND (NOT (e f));

1B6B2E Finally, the AND gives (a AND b) OR ((c/(-d)) AND (NOT (e f)));.

1B7 Dates and Initials

1B7A The (invisible) dates and initials attached to each statement may be tested with the constructs .SINCE, .BEFORE, .INITIALS=, and .INITIALS#. (The symbol # is used to mean "not equal".)

1B7B The .INITIALS construct requires the following format:

1B7B1 .INITIALS=ABC where the string abc is a user's initials (three initials must be given).

1B7C The .SINCE and .BEFORE constructs require the following format:

1B7C1 .SINCE (68/10/12 13:14) where 68 is the year, 10 is the month, 12 is the day, 13 is the hour, and 14 is the minute. The time may be eliminated by using 0:0.

1B7C2 Examples

1B7C2A .BEFORE (67/3/22 15:15) AND .SINCE (67/1/12 12:00); This pattern will cause display of only those statements bearing dates between noon of 12 January 1967 and 3:15 PM of 22 March 1967.

1B7C2B .SINCE (68/10/10 0:0) AND .INITIALS#DGC; This pattern will cause display of only those statements bearing dates later than 10 October 1968 and not bearing the initials DGC.

1B8 The WITHIN Construct

1B8A The WITHIN construct has the following format:

WITHIN n FIND exp1 SKIP exp2

where exp1 and exp2 are patterns and n is an integer. The search starts at the current position, and the content specified by exp2 is skipped up to n times in a search for the content specified by exp1. If any content other than what is specified by exp2 or exp1 is found, the search fails.

1B8A1 Example

1B8A1A ["write"] WITHIN 3 FIND " file" SKIP 1$NP1$PT; This pattern specifies the word "write" followed by the word "file", with up to three words intervening.

1B8A1A1 The search works as follows: after the word "write" is found, the search pointer indicates the space following the word. The exp2 pattern calls for one or more nonprinting characters followed by one or more printing characters; thus the space and the next word are skipped and the pointer again indicates a space. This skipping process is repeated up to three times, until the word "file" is found.

1B9 Special Control of Search

1B9A The position of the search pointer can be stored and set, and the direction of search can be controlled, in order to achieve complex effects. These effects also involve the use of the IF construct (described further on), and the possibilities have been explored only superficially at present. It should be possible to create pattern expressions of great complexity which would resemble sophisticated data-processing or information-retrieval programs, but at present the techniques

have not been worked out.

1B9A1 The position of the pointer may be stored in any one of nine buffers, P1 ... P9. This is done by writing ↑Pn, where n is some digit from 1 to 9.

1B9A1A The stored value in the buffer can then be decremented by writing ←Pn. The reason for doing this is that when the analyzer has found some entity, the pointer is moved to the next character position; in order to store the value of the last character actually searched, then, it is necessary to write ↑Pn←Pn.

1B9A2 The search pointer can then be set to the value in a buffer by writing Pn.

1B9A3 The search pointer can also be set to the beginning or end of a statement by writing SF(Pn) for the beginning and SE(Pn) for the end.

1B9A3A Note that SF and SE are functions which require a buffer value as argument; buffer values are not reinitialized after a statement has been scanned but continue to indicate the same character in the statement they were originally set to. Thus it is possible for a search to cover more than one statement.

1B9A4 The normal direction of scanning may be reversed by writing a less-than sign (<) and returned to the forward direction by writing a greater-than sign (>).

1B9A4A The left-arrow (←) used for decrementing a buffer value will increment it instead if the current scan direction is backward. Thus the effect will always be the same -- the buffer value will indicate the character just scanned.

1B9A4A1 Example

1B9A4A1A ↑P1 SE(P1) < $NP -'.; This pattern causes statements to be searched backwards from the end. Only statements whose last printing character is not a period will be displayed.

1B9A4A1A1 The construct "↑P1" at the beginning of the pattern causes the current pointer position (which indicates the beginning of the statement) to be stored. This is simply for the purpose of having an argument for the "SE(P1)" construct, which causes the pointer to be positioned to the end of the statement. The less-than sign then causes the scan to proceed backwards; any number of nonprinting characters

will be permitted, and then a character which is not a period is specified.

1B10 The IF Construct

1B10A The IF construct has the following format:

(IF relat THEN exp1 ELSE exp2)

where "relat" is a relationship between two buffer values and exp1 and exp2 are pattern expressions.

1B10B The possible relationships are as follows:

1B10B1 .EQ (equals)

1B10B2 .NE (not equal to)

1B10B3 .LT (less than)

1B10B4 .LE (less than or equal to)

1B10B5 .GT (greater than)

1B10B6 .GE (greater than or equal to).

1B10C If the specified relationship is true, exp1 is used for a test; if it is not true, exp2 is used.

1B10C1 Example

1B10C1A ↑P1 SE(P1) < (['e] ↑P2←P2 AND ['t] ↑P3←P3) (IF P2 .LT P3 THEN SF(P1) > $SP "The" ELSE SF(P1) > [" if "]); This pattern imposes the following condition on statements to be displayed: If the last "e" precedes the last "t", then the first word in the statement must be "The". Otherwise, the statement must contain the word "if", enclosed by spaces. The proof is left to the reader.

1B11 The .EMPTY Construct

1B11A Whenever the analyzer makes a test, a flag is set true or false. After a statement has been tested by the complte pattern, it is displayed if the flag is true and omitted if the flag is false.

1B11B The construct .EMPTY simply sets the flag true. Conversely, the construct NOT .EMPTY (or -.EMPTY) sets the flag false.

1B11C This is useful in the IF construct, where one may simply wish to test the relationship without imposing further tests.

1B11C1 Example

1B11C1A ↑P1 SE(P1) < (['e] ↑P2←P2 AND ['t] ↑P3←P3) (IF P2 .LT P3 THEN NOT .EMPTY ELSE .EMPTY); This pattern is similar to the previous example, but slightly simpler. The condition is that if the last "e" in the statement does not precede the last "t", the statement will be displayed, otherwise it will not.

1B11C1A1 The "↑P1" stores the pointer value, which inicates the beginning of the statement. The "SE(P1)" sets the pointer to the end of the statement, and the "<" causes a backward scan. An "e" is found and its position stored in P2; then a "t" is found and its position stored in P3. The IF construct compares the values of P2 and P3: if P2 is less than P3 (i.e. if the "e" precedes the "t" in the statement), the "NOT .EMPTY" takes effect and the flag is set false, so the statement will not be displayed; if P2 is not less than P3, the ".EMPTY" takes effect, the flag is set true, and the statement is displayed.

1C Procedure for Using Content Analyzer

1C1 A pattern may be written as text anywhere in a file. A file may thus contain any number of patterns; however only one pattern may be compiled at a time -- i.e. when a new pattern is compiled the code created by the previous one is lost.

1C2 To compile a pattern, the command Execute Content Analyzer is used. The syntax is

ec [c1] CA

where [c1] means that a character is selected either with the mouse or by means of a pointer call, and CA means that a Command Accept key is struck.

1C2A The character selected must be either the first character of the pattern or a nonprinting character preceding the pattern, with no printing characters intervening.

1C2B Note that the last part of a pattern may thus be used as a separate pattern, if it is meaningful.

1C3 The screen will go momentarily blank with a message. If the pattern has been compiled, the message is "successful compilation"; if the pattern has an error in it which prevents it from compiling, the message is "syntax error".

1C3A Syntax errors are fequently caused by inadvertent omission of some character such as a quotation mark. Another common cause for a syntax error or a compiled pattern that does

not work as expected is an error in the way parts of the pattern are grouped. In the latter case, the problem may often be solved by insertion of parentheses.

1C4 When the pattern has been compiled, it will not go into effect until the view-control parameter "i" is placed in effect. When this has been done, the system will display only statements which fit the pattern.

1C5 Testing of statements begins with the statement currently designated as the display start; other statements are then tested in the order in which they would appear "normally," i.e. with the analyzer off. Any other view specifications which are in effect continue to work; thus if only first- and second-level statements are being displayed, only first- and second-level statements will be tested by the analyzer.

1C6 Statements are tested until the display screen has been filled. If no statements are found that fit the pattern, the screen goes blank with the message "empty" and remains so until the analyzer is turned off or until changed view-control parameters make it possible to find a statement that fits the pattern.

1C7 Whenever the display is recreated, the testing process is repeated. Thus if a statement is edited, and the editing changes it so that it no longer fits the pattern, it will disappear from the screen.

LINK SYNTAX AND USEFUL KNOWLEDGE

Preliminary Document on Links

Link syntax: (username,filename,statementdes:VIEWSPECs)

The username is the name of the user under whose name the desired file is stored on the RAD.

If it is omitted, the name of the owner of the currently displayed file is assumed. Note that this is not necessarily the same name that has been given to the Exec or to NLS when entering, nor the name under which the current file is stored on the RAD.

The filename is the name under which the desired file is stored on the RAD.

If this is omitted, the currently displayed file is assumed. Note that in this case a username may not be given.

The statementdes is either a statement name or a statement number.

If this is omitted, the origin of the file is assumed.

The VIEWSPECs are the usual code letters, in a string just as if they were being entered from the keyboard or keyset.

If this is omitted, the current VIEWSPECs are assumed.

The Intrafile Return Ring

Whenever any jump is made within the file (except by Jump Return and Jump Ahead), a new entry is made in a list called the intrafile ring.

Each of these entries gives a display start and a set of VIEWSPECs.

A "pointer" indicates the current view on the list; each time a normal jump is executed, the new information is written ahead of the pointer and the pointer is moved forward.

On a Jump Return or Jump Ahead command, the pointer is simply moved backward or forward and no new entries are made.

The ring may hold a maximum of five entries. If a Jump Ahead is made from the last entry, the pointer is moved to the first entry; conversely, if a Jump Return is made from the first entry the pointer is moved to the last entry (hence the term "ring").

The Interfile Return Stack

The interfile stack works much like the intrafile stack, with the following exceptions and complications:

The length of the list is variable, and depends on the amount of information in the links used.

The list is not set up as a ring; a Jump to File Ahead from the last entry or Jump to File Return from the first will cause an error message and abort the command.

The WORKING COPY file is never entered in the stack (the WORKING COPY is explained below).

The CHECKPOINT file is never entered in the stack.

A new entry is made on the stack whenever a file is loaded with the Load File command (not Load Checkpoint), or whenever any interfile jump is executed with Jump to Link or Jump to File Link.

A pointer indicates the current view, just as in the intrafile ring.

The command Jump to File Working Copy causes the WORKING COPY file to be opened and displayed; the pointer is not moved, but continues to indicate the last file viewed in the stack. The command Jump to File Current causes this file to be opened and displayed.

Opening and Displaying Files -- The WORKING COPY File

When a file is loaded or jumped to, it is "opened" and displayed; no copy is created, rather the file itself is viewed directly on the RAD. However, if any changes are made in the file it becomes impossible to continue this. At some point the system will create a WORKING COPY, copy the displayed file to it, close the displayed file, and display the WORKING COPY instead, with the changes.

The WORKING COPY is not always made as soon as changes are specified; however, when it is created a message to that effect appears on the screen.

Creation of a WORKING COPY can always be forced by specifying the command Output File and then hitting CO instead of CA.

Creation of a WORKING COPY can always be avoided by doing nothing except intrafile jumps and VIEWSPECs.

When you create a new file, working in NLS, the file is the WORKING COPY (no message is displayed in this case to show creation of WORKING COPY).

When you output such a file with Output File, it is NOT entered

in the interfile stack. To enter it on the stack, you must load it or jump to it.

You can, however, display it with Jump to Working Copy, unless a new WORKING COPY is created from some other file.

Special Points and Helpful Hints

It is possible to move the pointer without displaying a new file. This will happen if you specify a Jump to File Ahead or Return and then abort it with CD. The results can be confusing but since the stack is never very long you can generally find your way back to where you wanted to go fairly quickly.