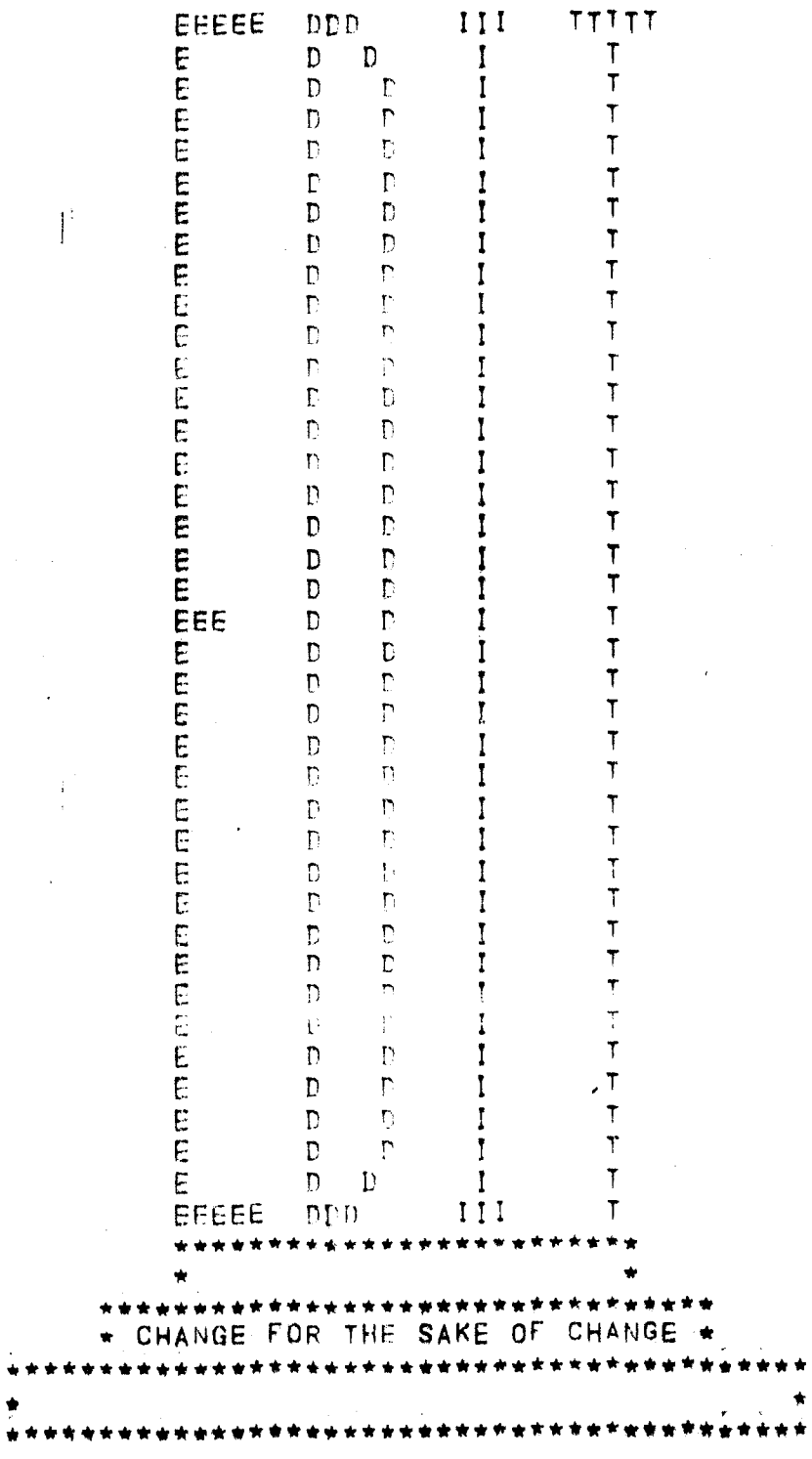
SETL NEWLETTER 126
A SETLB SPECIFICATION OF EDIT

MICHAEL BRENNER
FEBRUARY 4,1974



A SETL SPECIFICATION OF EDIT

- MICHAEL F BRENNER

ABSTRACT

THIS THESIS GIVES A SPECIFICATION OF A GENERAL PURPOSE TEXT EDITOR IN SETLB, A VERY HIGH LEVEL LANGUAGE. THE SYNTAX OF THE SETL EDITOR IS A SUPERSET OF THE SYNTAX OF THE NYU EDITOR WHICH HAD BEEN WRITTEN IN COMPASS, THE ASSEMBLY LANGUAGE FOR THE CDC 6600 COMPUTER. A KNOWLEDGE OF THE NYU EDITOR IS ASSUMED AND A COPY OF ITS DOCUMENTATION IS INCLUDED AS AN APPENDIX. THE SETL EDITOR ADDS A GENERALIZED TREATMENT OF GLOBAL VARIABLES, MORE PRIMITIVES, AND ABOVE ALL TOTAL FREEDOM FROM REGISTER ALLOCATION. IT WILL SERVE AS A BASIS FOR AN EVENTUAL IMPLEMENTATION IN A LOWER LEVEL LANGUAGE.

KEYWORDS

EDIT
HIGH LEVEL LANGUAGE
NYU EDITOR
SETL
SPECIFICATION
STRING MANIPULATION
TEXT EDITOR
TOP DOWN PROGRAMMING

THIS PROGRAM WAS WRITTEN USING COMPUTER TIME AND OTHER FACILITIES GRANTED BY THE SETL GROUP AT THE COURANT INSTITUTE OF MATHEMATICAL SCIENCES AT NYU, AEC COMPUTER CENTER.

TABLE OF CONTENTS:

THE NYU EDITOR IS AN ADVANCED, EFFICIENT, AND EASY TO USE EXAMPLE OF A MODERN GENERAL PURPOSE TEXT EDITOR. IT CAN GREATLY REDUCE THE EFFORT REQUIRED TO CREATE, MANIPULATE, AND MODIFY FILES CONTAINING CARD IMAGES OR OBJECTS RESEMBLING CARD IMAGES. IT WAS WRITTEN BY RICHARD REICH AND IMPROVED BY MICHAEL RYAN AND WILLIAM RUSSELL WHO IS CURRENTLY MAINTAINING IT. MICHAEL RYAN WROTE CHAPTERS 1 TO 4 OF THE DOCUMENTATION INCLUDED HERE AS AN APPENDIX.

THOSE WHO HAVE USED THE NYU EDITOR AT NYU ARE ALL GRATEFUL FOR THE TIME IT HAS SAVED THEM. LIKE TELEVISION OR SUBWAYS, ONE WONDERS HOW WE COULD GET ALONG WITHOUT IT FOR SO LONG.

/* 1. INTRODUCTION */

/* 1.1 WHAT IS A TEXT EDITOR */

COMPUTER PROGRAMMERS CREATE AND MODIFY COMPUTER PROGRAMS WHICH FROM AN ABSTRACT POINT OF VIEW CONSIST OF NOTHING BUT CHARACTER STRINGS (TO BE SURE, CHARACTER STRINGS INSTILLED WITH A LARGE AMOUNT OF THE THEOLOGICAL VIRTUE OF HOPE). TO REDUCE THE EFFORT REQUIRED TO MANIPULATE THESE CHARACTER STRINGS IS TO INCREASE THE PRODUCTIVITY AND SATISFACTION OF PROGRAMMERS.

IN THE EARLY DAYS OF COMPUTER SCIENCE, THE PROGRAMMER HAD ONLY ONE WAY TO MANIPULATE THE SOURCE TEXT OF HIS PROGRAMS, THE KEY PUNCH, WITH THE TREMENDOUS ADVANCES MADE BY THE ART SINCE THEN, WE NEED MORE THAN A FASTER KEY PUNCH, EVEN MORE THAN ONE WITH A BUFFER FOR ON-LINE CORRECTION OF TEXT. WE NEED SOMETHING WITH JUST A SLIGHT BIT OF INTELLIGENCE IN THE SENSE THAT IT CAN UNDERSTAND WHAT WE WANT IT TO DO AND THEN DO IT.

FROM THESE NEEDS EVOLVED THE IDEA OF TEXT-EDITOR, A COMPUTER PROGRAM FOR CREATING AND MODIFYING SOURCE PROGRAMS, USING THE COMPUTER ITSELF TO DO THE HARD PART OF THE WORK. BY MAKING MOST OF THE EFFORT PREVIOUSLY EXPENDED BY THE PROGRAMMER AUTOMATIC, THE MANIPULATION OF PROGRAMS BECAME EASIER, FASTER, MORE DEPENDABLE AND LESS ERROR PRONE.

WE SHOULD EXAMINE THE CHARACTERISTICS OF A GENERAL PURPOSE TEXT EDITOR. WHAT, FOR INSTANCE, IS THE DIFFERENCE BETWEEN A TEXT EDITOR AND AN AUTOMATIC UPDATE PROGRAM SUCH AS CDC'S UPDATE. AN UPDATE PROGRAM MAINTAINS LIBRARIES OF SOURCE PROGRAMS BY INSERTING, DELETING, AND MOVING CARD IMAGES, AND A CARD IMAGE IS THE ATOM, OR UNIT OF THE SYSTEM. WHILE INSERTING, DELETING AND MOVING ARE DONE QUITE EFFICIENTLY, THE INDIVIDUAL CHARACTER OR CARD COLUMN BECOMES THE ATOMIC UNIT TO BE MANIPULATED BY AN EDITOR. THUS UPDATE PROGRAMS WORK ONE LEVEL HIGHER (WITH CARDS AND SOMETIMES WHOLE FILES) THAN

EDITORS WHICH WORK WITH CHARACTERS AND SOMETIMES WITH CARDS.

THE UPDATE PROGRAMS NUMBER THEIR CARDS IN A PERMANENT MANNER SO THAT CARD IDENTIFIERS DO NOT CHANGE EVEN AFTER CORRECTIONS HAVE BEEN APPLIED. THE NYU EDITOR WAS IMPLEMENTED WITH A DIFFERENT SET OF GOALS IN MIND. AS A MATTER OF DECISION, THERE ARE NO CARD IDENTIFIERS OTHER THAN THE CARDS CURRENT POSITION ORDINAL IN THE FILE. IF A CARD IS INSERTED, THE ORDINALS OF ALL CARDS AFTER IT ARE INCREMENTED BY ONE.

THE BASIC MANIPULATION A TEXT EDITOR PERFORMS ON CHARACTER STRINGS IS CHANGING ONE STRING TO ANOTHER. THE COMMAND LANGUAGE MUST BE ESPECIALLY GEARED TO HUMAN EASE OF USE, CONCISENESS, AND A LARGE AMOUNT OF EXPRESSIVITY FOR A SMALL NUMBER OF CHARACTERS TYPED, FOR EDITORS ARE MOSTLY USED INTERACTIVELY. IN SITUATIONS WHERE BOTH THE TYPIST AND THE COMPUTER HAVE BETTER THINGS TO DO THAN EXCHANGE LONG CONVERSATIONS WITH EACH OTHER TO DECIDE WHAT EACH ONE WANTS.

/* 1.2 TEXT EDITOR VS. PROGRAMMING LANGUAGE */

THE NEXT STEP AFTER THIS EDITOR SEEMS TO BE AN INTERACTIVE IMPLEMENTATION OF THE SNOBOL LANGUAGE WHICH IS UNDESIRABLE BECAUSE

IT IS EXTREMELY INEFFICIENT IN SNOBOL TO DO THE SIMPLE CHORES AN EDITOR NEEDS TO DO ESPECIALLY A LOT OF INPUT/OUTPUT. (A TEXT EDITOR IS VERY I/O BOUND)

ALONG THESE LINES, THE IMPORTANT QUESTION TENDS TO ARISE WHEN DESIGNING SYSTEMS OF THIS TYPE, EXACTLY WHERE DOES THE GENERAL PURPOSE TEXT EDITOR LEAVE OFF AND A STRING MANIPULATION PROGRAMMING LANGUAGE BEGIN. WE CAN NOT COME UP WITH A DECISIVE ANSWER, BECAUSE OF THE LACK OF A COMPLETE DEFINITION OF AN EDITOR, SURELY ALL WILL AGREE THAT AN EDITOR IS MEANT TO BE SOMEWHAT LESS THAN A PROGRAMMING LANGUAGE. NEVERTHELESS, THERE WILL ALWAYS BE USERS OF THE EDITOR WHO WANT NEW FEATURES CONSTANTLY BEING ADDED TO THE EDITOR, AND OTHERS WHO USE SNOBOL OR SIMILAR ADVANCED PROGRAMMING LANGUAGES, WHO WILL CRY OUT FOR MORE EFFICIENCY IN THEIR PROGRAMS.

EDITORS AND PROGRAMMING LANGUAGES HAVE IN COMMON THAT THEY REACH INTO A CHARACTER STRING (OR PERHAPS A COMPLEX DATA STRUCTURE) AND DETERMINE WHETHER IT IS OF A CERTAIN TYPE (THAT IS, WHETHER IT FULFILLS THE CRITERA OF CERTAIN BOOLEAN PREDICATES) AND PICK OUT THE SUBSTRING WHICH CAUSED IT TO FULFILL THE CRITERIA, EFFICIENCY VS. GENERALITY FAIRLY CHARACTERIZES THE DIFFERENCES BETWEEN EDITORS AND PROGRAMMING LANGUAGES AND WILL BE THE DETERMINING FACTOR FOR ANY SYSTEMS ANALYST CONTEMPLATING IMPLEMENTATION OF SUCH SYSTEMS.

IN A GENERAL STRING MANIPULATION LANGUAGE, THE FULL POWER OF RECURSIVE PATTERN MATCHING IS TYPICALLY IMPLEMENTED IN SUCH A WAY THAT PATTERNS SPECIFIED BY ANY CONTEXT-FREE GRAMMARS CAN BE MATCHED IN A FINITE AMOUNT OF TIME AND ANY GRAMMAR OR FRAGMENT OF A GRAMMAR CAN INITIATE A SEARCH PROCESS. SUCH FRAGMENT-INITIATED SEARCHES WILL PROBABLY TERMINATE BECAUSE THE PROGRAMMER HAS CHOSEN A SUITABLE CONTEXT AND A SUITABLE PATTERN, ALTHOUGH WITH ARBITRARY INPUT THERE IS NO GUARANTEE OF TERMINATON OR IN THE CASE OF TERMINATION IN A FINITE AMOUNT OF TIME, THERE IS NO GUARANTEE OF MEANINGFUL RESULTS.

AS A MATTER OF PRESCRIPTIVE DEFINITION, THEN, A PROGRAMMING LANGUAGE CAN MATCH PATTERNS SPECIFIED BY ANY CONTEXT FREE GRAMMAR. IT IS USED WHEN COMPLETENESS IS A DEFINITE REQUIREMENT AND EFFICIENCY IS NOT AS IMPORTANT AS COMPUTING POWER.

IN AN EDITOR THE SEARCH IS STRICTLY A TREE-SEARCH AS OPPOSED TO A GRAPH SEARCH, WITH BACKUP EITHER BEING NONEXISTANT OR EXTREMELY RARE. THE TYPE OF TREES TYPICALLY GENERATED BY EDIT PATTERNS ARE ONE-LEVEL VECTORS OF ELEMENTARY PATTERNS WHICH ARE NEVER RECURSIVE SINCE THEY CONTAIN NO POINTERS TO OTHER PATTERNS. NEVERTHELESS, WHILE THEORETICALLY INCOMPLETE THEMSELVES, EDITORS ARE EXTREMELY USEFUL TO CREATE AND UPKEEP COMPLETE PROGRAMMING LANGUAGES.

EDITORS ARE MUCH MORE EFFICIENT IN USING THE COMPUTERS RESOURCES OF TIME AND MEMORY SPACE THAN THE CORRESPONDING COMPLETE SYSTEMS. NEVERTHELESS, THE KEY ADVANTAGE OF EDITORS OVER PROGRAMMING LANGUAGES IS NOT BASICALLY THAT THEY ARE EFFICIENT FOR THE COMPUTER, BUT THAT THEY ARE EFFICIENT FOR THE PROGRAMMER, SINCE EDITORS ARE USUALLY INTERACTIVE, AND SINCE THE DATA STRUCTURES ARE EVEN SIMPLER THAN MOST LOW LEVEL LANGUAGES, ALLOWING ONLY CHARACTER STRINGS AS DATA TYPES. (BUT INTERPRETING CERTAIN CHARACTER STRINGS AS PATTERNS), DEBUGGING THE NECESSARILY SIMPLE PROGRAMS IS MUCH EASIER AND THEREFORE FASTER THAN DEBUGGING CORRESPONDING RECURSIVE ROUTINES, EVEN FASTER THAN ROUTINES IN LANGUAGES WITH AUTOMATIC RECURSION CONTROL, SUCH AS PL/I, BALM, OR SNOBOL4.

/* 1.3 SOURCE LANGUAGE */

SETL WAS CHOSEN AS THE LANGUAGE OF IMPLEMENTATION FOR A NUMBER OF REASONS. FOR ONE THING, SETL IS A VERY HIGH LEVEL SPECIFICATION LANGUAGE YET STILL EXECUTABLE. IT WAS DESIGNED TO REMOVE FROM THE PROGRAMMER ANY CONSIDERATION OF DATA STRUCTURE OR MACHINE DEPENDENCIES. ITS SYNTAX IS CLOSE TO PURE MATHEMATICAL NOTATION. SETL IS MACHINE INDEPENDENT AND DATA STRUCTURE INDEPENDENT IN THE SENSE THAT ANY DATA STRUCTURE IMPLEMENTED TO MAP SETL DATA OBJECTS ONTO HARDWARE FEATURES IN A PARTICULAR IMPLEMENTATION MAY BE CHANGED TO ANY OTHER DATA STRUCTURE WITHOUT SUBSTANTIALLY MODIFYING THE SETL ALGORITHM, AND PROBABLY WITHOUT MODIFYING IT AT ALL. THIS SEPARATION OF THE ABSTRACT ALGORITHM FROM THE UNDERLYING DATA STRUCTURE RESULTS IN VERY UNDERSTANDABLE AND USEFUL SPECIFICATIONS. IT IS THUS EASIER TO TRANSPORT SETL PROGRAMS ACROSS COMPUTER BOUNDARIES THAN ASSEMBLY PROGRAMS.

AS AN EXAMPLE OF POSSIBLE DETRIMENTAL PROGRAMMER CONSIDERATION OF DATA STRUCTURE, WE MAY CONSIDER THE REPRESENTATION OF CHARACTER STRINGS THEMSELVES. THE INPUT AND OUTPUT OF THE STRINGS WILL BE DONE IN SOME SORT OF PACKED MODE, THAT IS, MORE THAN ONE CHARACTER WILL BE STORED IN EACH COMPUTER WORD. HOWEVER, FOR THE PATTERN MATCHING, IT IS MUCH TOO COMPLEX TO DO IT IN PACKED MODE, AND THE STRING WILL BE TRANSFORMED INTO AN UNPACKED FORMAT WITH ONLY ONE CHARACTER PER WORD (OR BYTE, AS THE CASE MAY BE). THE HIGH LEVEL LANGUAGE CAN DO THIS WITH THE BEST OF THE TWO METHODS, NAMELY ALWAYS STORING THE STRINGS IN PACKED MODE FOR EFFICIENCY, BUT GIVING THE PROGRAMMER THE SYNTACTIC ILLUSION THAT HE HAS THE STRINGS IN UNPACKED MODE. THE ALGORITHMS ARE THUS SIMPLER AND PROBABLY BETTER WRITTEN THAN THEY WOULD BE IN A LOWER LEVEL LANGUAGE.

A PROBLEM WITH THE NYU EDITOR THAT WOULD NEVER SHOW UP IN A HIGHER LEVEL LANGUAGE IS THE DECISION OF HOW TO REPRESENT THE CARD IMAGES AND FILES AND RECORDS ON THE DISK. THE CHOICES ARE INFINITE. THE NYU EDITOR ENDS EACH LINE WITH DOUBLE ZERO CHARACTERS, BUT IF THE FULL CHARACTER SET IS GOING TO BE USED FOR TEXT, THEN SOME METHOD MUST BE ARRIVED AT TO ASSURE THAT A PACKED WORD OF TEXT (SAY, EVERY 10 CHARACTERS), DOESNT END WITH TWO CONSECUTIVE ZERO CHARACTERS. DOUBLE BUFFERING AND CIRCULAR INPUT/OUTPUT BUFFERS ARE COMBINED IN THE NYU EDITOR FOR EFFICIENCY, WITH THE CORRESPONDING LOSS OF READABILITY. HIGHER LEVEL LANGUAGES WOULD ALSO BE FREE OF WAITING PRIMITIVES, SUCH AS WAIT FOR INPUT-OUTPUT COMPLETION. THE ≠OPERATING SYTEM≠ IS ASSUMED TO TAKE CARE OF WAITING FOR I/O COMPLETION WHENEVER AN I/O COMMAND IS REQUESTED ON A DEVICE THAT IS STILL BUSY FROM A PREVIOUS REQUEST. THE USER WILL NEVER BE AWARE OF THIS, AND HIS ALGORITHM WILL NEVER BE ≠DIRTIED≠ BY

IT TO TAKE ADVANTAGE OF SUCH OPERATING SYSTEM PRIMITIVES AS A NON-STOP READ ROUTINE.

STRANGELY ENOUGH, THE VERY SEMANTIC FEATURES WHICH ARE RESPONSIBLE FOR THE BIRTH OF SETL ARE NOT THE FEATURES WHICH CAUSED IT TO BE CHOSEN AS THE SPECIFICATION LANGUAGE FOR THE EDITOR. SETS ARE USED ONLY AS FAST REPRESENTATIONS OF LINEAR LISTS, AND THE ONLY DATA STRUCTURES NEEDED ARE THESE LINEAR LISTS (DOUBLE ARRAYS -- IN SETL, A SET OF 2-TUPLES OF WHICH THE HEAD AND TAIL ARE BOTH CHARACTER STRINGS), CHARACTER STRINGS, AND INTEGERS WHICH ARE ALL INTEGERS IN THE HARDWARE INTERPRETABLE OBJECT CODE AND ALL CHARACTER STRINGS AS FAR AS EDIT KNOWS.

SETL WAS CHOSEN BECAUSE DESPITE A CERTAIN INCOMPLETENESS OF ITS OWN SPECIFICATION (THAT IS, THE SETL SPECIFICATION OF SETL, CF KURT B MOLYS SETL NEWLETTERS 19, 44, 47, 50, 58, 61, 62, AND 64 FOR THE SETL BOOTSTRAP COMPILER.
ITS SYNTAX IS AT SUCH A HIGH LEVEL THAT, (IN ACCORDANCE WITH THE PURPOSES OF THE EDITOR ITSELF),IT IS POSSIBLE TO DIRECT THE MACHINE TO DO A TASK USING LESS EFFORT ON THE PART OF THE PROGRAMMER.

IN ORDER TO BETTER REALIZE THE SUBGOALS OF TRANSPORTABILITY AND READABILITY, CERTAIN #GOOD PROGRAMMING PRACTICES# HAVE BEEN FOLLOWED IN THIS PROGRAM. ALL MACROS APPEAR IN THE MACRO SECTION NEAR THE BEGINNING OF THE PROGRAM. (SECTION 3.2). ALL GLOBAL VARIABLES ARE INITIALIZED IN AN INITIALIZATION SECTION (SECTION 3.3). ALL LOCAL VARIABLES ARE INITIALIZED BEFORE USING THEM.

FINALLY, THIS HIGH LEVEL SPECIFICATION WILL SERVE AS A MODEL TO IMPLEMENT AN EDITOR IN A LOW LEVEL LANGUAGE SUCH AS PL/I OR PL/STAR, ASSEMBLY CODE NEVER HAS TO BE USED IF WE ASSUME A VERY BASIC OPERATING SYSTEM WHICH PROVIDES THE EFFICIENT INPUT AND OUTPUT JUST MENTIONED. THE ASSEMBLY CODED NYU EDITOR IS A LEVEL MORE COMPLEX THAN NEED-BE BECAUSE OF HAND OPTIMIZATION OF THE INDIVIDUAL COMMANDS. THE CODE GENERATED BY AN OPTIMIZEABLE LOWER LEVEL LANGUAGE WITH AUTOMATIC REGISTER ALLOCATION WILL ALLOW THE SOURCE CODE TO MAINTAIN ITS READABILITY WHILE THE BINARY WILL BE ABOUT AS GOOD AS THE HAND WRITTEN CODE.

/* 1.4 JOB CONTROL LANGUAGE */

THIS ENTIRE DISCUSSION OF THE EDITOR IS ≠EXECUTABLE≠ ON THE CDC 6600 COMPUTER. THERE ARE ACTUALLY THREE DIFFERENT LEVELS INCLUDED IN THE PROGRAM LIBRARY CONTAINING THE CARD IMAGES FOR THIS PROGRAM, DETERMINED BY THE DOCUMENT KEY, THE TESTING KEY OR NO KEY. ASSUMING ALL REFERENCED FILES EXIST IN THE SYSTEM ALREADY, THE FOLLOWING SETS OF JOB CONTROL STATEMENTS WILL WORK UNDER SCOPE 3.2, 3.3, OR 3.4 OPERATING SYSTEMS.

WHAT YOU ARE READING NOW WAS PRODUCED BY THE FIRST SET OF JCL BY DEFINING THE KEY ≠DOCUMENT≠ WHICH ADDS COMMENTS TO THE PROGRAM. ORIGINALLY THESE COMMENTS WERE INCLUDED IN THE ≠TEST≠ KEY BUT LEXICAL SCANNING TIME WAS REDUCED BY PRINTING OUT THE EXTENDED COMMENTS ONLY WITH DOCUMENTED LISTINGS. EXTENDED COMMENTS ARE SIGNIFIED BY LINES WHOSE FIRST TOKEN IS A DOLLAR SIGN.

BY NOT DEFINING ≠DOCUMENT≠ NOT ONLY ARE THE EXTENDED COMMENTS REMOVED BUT ALSO SOME OPERATORS ARE DEFINED WHICH WILL BE INCLUDED IN THE RUN-TIME LIBRARY OF THE NEXT VERSION OF THE SETL COMPILER, BUT WHICH MUST BE HARD DEFINED DURING EACH SETL RUN IN THE PRESENT COMPILER. THEY ARE [illegible]. SEE JOB CONTROL LANGUAGE SET NUMBER TWO BELOW. THE ENTIRE SYSTEM IS NOW SET UP FOR EXECUTION, ASSUMING THAT THE MACHINE WERE BIG ENOUGH.

SINCE THE NYU SETL COMPUTER, A CDC 6600, IS NOT BIG ENOUGH (HAVING ONLY 400K OCTAL OF MEMORY), ANOTHER OPTION IS AVAILABLE TO AVOID THE DIFFICULTIES OF HAND-MANAGED MEMORY ALLOCATION, NAMELY TO DEFINE THE KEY ≠TEST≠. THIS DEFINITION ALLOWS INDIVIDUAL SUBROUTINES OR GROUPS OF SUBROUTINES TO BE TESTED BY PAGING THEM INTO MEMORY WHENEVER CALLED. SEE JCL SET 3 BELOW.

THIS KIND OF MULTI-LEVEL COMMENTING IS EXTREMELY USEFUL AND SHOULD BE CONSIDERED AN ESSENTIAL PART OF ANY LANGUAGE BEING DEVELOPED. THE CDC UPDATE PROGRAM SUPPORTS IT, WITH NOT TOO MUCH TROUBLE TO THE USER, ESPECIALLY IF THE UPDATE DIRECTIVES ARE MANAGED WITH A GENERAL PURPOSE TEXT EDITOR AS THIS PROGRAM WAS. BUT A LANGUAGE ITSELF SHOULD SUPPORT CONDITIONAL DISPLAY OF COMMENTS (AND IN THE SAME REGARD, CONDITIONAL ASSEMBLY OF CODE) IN A MANNER WHICH IS ACTUALLY CONVENIENT AND EASY TO USE. SETL, EVEN IN THE PRELIMINARY IMPLEMENTATION RUNNING UNDER THE SCOPE 3.2 INCOMPLETE OPERATING SYSTEM, AFFORDS CONDITIONAL DISPLAY OF COMMENTS BY ELIMINATING (AT THE USERS OPTION) ANY LINE BEGINNING WITH A DOLLAR SIGN. IN THE CODE COMPILED HERE I HAVE CHOSEN TO USE THE UPDATE PRE-PROCESSOR TO THE SETL COMPILER BECAUSE HAVING DOLLAR SIGNS IN COLUMN 1 PRECLUDES HAVING PAGE EJECT CHARACTERS IN COLUMN 1 FOR COMMENTS, WHICH BECOMES IMPORTANT WHEN THE TEXT BECOMES VERY LONG, AS IN THIS PROGRAM. THUS THE TWO SETS OF DOCUMENTATION MANAGEMENT COMPLEMENT EACH OTHER HERE.

JCL SET I. TO GET A NICELY FORMATTED LISTING WITH EXTENDED COMMENTS
(NOTE -- THE COMPILE FILE IS SUITABLE FOR INPUT
TO THE NYU MANUALMAKER PROGRAM.

```
ID, CM 34000, T 2, NAME
ATTACH(OLDPL,SETLMANUALSPL)
UPDATE(C=OUTPUT)
   *** END-OF-RECORD CARD ***
*DEFINE DOCUMENT
*C EDITSETL
   *** END-OF-FILE CARD ***
```

JCL SET II. TO GET A RUN OF THE ENTIRE SYSTEM

```
ID, CM 600000, T 77777. NAME
ATTACH(OLDPL,SETLMANUALSPL)
UPDATE.
SETLB(COMPILE) THE SETLB LEXICAL SCANNER
ATTACH(BL=4SVD,SAVESETL) GET SETL SAVE FILE
BALM(SETLOUT)
   *** END-OF-RECORD CARD ***
*C EDITSETL
   *** END-OF-FILE CARD ***
```

JCL SET III. TO TEST SUBROUTINES OR MODIFICATIONS OF SUBROUTINES

```
ID, CM 340000, T 3000. PRERUN
ATTACH(OLDPL,SETLMANUALSPL)
UPDATE.
SETLB(COMPILE)
ATTACH(BAL=,BALMTEMP)
ATTACH(BL=4SVD,SAVESETL)
BALM(SETLOUT)
   *** END-OF-RECORD CARD ***
*C EDITSETL
*DEFINE TEST
(ANY OTHER CORRECTIONS TO BE TESTED)
   *** END-OF-FILE CARD ***
```

/* 2.1 CONTROL STRUCTURE */

AN EDIT SESSION IS AN N-TUPLE OF TELETYPE LINES TRANSMITTED TO THE COMPUTER. EACH LINE IS COMPOSED OF A NUMBER OF EDIT COMMANDS, SYSTEM COMMANDS, OR MACROS, SEPARATED BY SEMICOLONS. A MACRO IS A AN N-TUPLE OF EDITOR COMMANDS, SYSTEM COMMANDS, AND OTHER MACROS, ALTHOUGH IN THE NYU EDITOR, AS CURRENTLY IMPLEMENTED AT NYU, MACROS MAY NOT RECURSIVELY INVOKE OTHER MACROS. EACH MACRO, SYSTEM COMMANDS, AND EDITOR COMMAND IS FORMATTED AS A KEYWORD FOLOWED BY A LIST OF ARGUMENTS. REPETITIONS AND LOOPS ARE PERMITTED IN THE EDITOR, BUT ONLY LIMITED TO THE LINE THAT IS CURRENTLY BEING EXECUTED. A GO TO DICTION IS AVAILABLE IN THE COMMAND SKIP, BUT IT IS ONLY POSSIBLE TO SKIP FORWARDS, MAKING A GRAPH STRUCTURED PROGRAM IMPOSSIBLE. THE DATA OBJECTS OF THE EDITOR INCLUDE STRINGS AND PATTERNS, BUT PATTERNS ARE JUST STRINGS. INTEGERS EXIST IN THE COMMAND LANGUAGE FOR THE PURPOSE OF ITERATION CONTROL AND SPECIFYING PATTERN FIELDS, BUT THE EDITOR DOES NOT INTERNALLY MANIPULATE OBJECTS OF TYPE INTEGER. THIS MEANS THAT MANY POSSIBLE SNOBOL PATTERNS CAN NOT BE IMPLEMENTED BY EDITOR MACROS. THIS DEFICIENCY IS SELDOM MISSED. THUS THE DECISION TO LEAVE OUT INTEGERS SEEMS TO HAVE BEEN BEST IN THE INTERESTS OF EFFICIENCY BUT FOR THE PURPOSES OF A GENERAL PROGRAMMING LANGUAGE THIS WOULD BE IMPOSSIBLE. THE BASIC CONCLUSION IS THEREFORE THAT IF YOU NEED PATTERNS DETERMINED BY DATA OBJECTS OF TYPES OTHER THAN STRINGS, THEN YOU PROBABLY NEED A COMPLETE PROGRAMMING LANGUAGE LIKE SNOBOL, AND EFFICIENCY WILL NOT BE THE MOST IMPORTANT CONSIDERATION.

MACROS ARE EXPANDED AT RUN TIME WITH DYNAMIC SUBSTITUTION OF ARGUMENTS. AT ANY POINT IN THE CONTROL STREAM, A MACRO MAY APPEAR. THIS RESULTS IN SOME POINTERS BEING SAVED AND RESET SO THAT ALL FURTHER COMMANDS COME FROM THE MACRO FILE UNTIL THE RECORD IN THE MACRO FILE CORRESPONDING TO THE MACRO BEING INVOKED IS DEPLETED. THEN CONTROL RETURNS TO THE NORMAL INPUT STREAM.

ALMOST EQUIVALENT TO A MACRO IS A READ FILE. THE READ FILE IS FASTER THAN MACRO EXPANSION BECAUSE NO ARGUMENT SUBSTITUTION IS PERMITTED IN THE READ FILE. OTHERWISE, IT IS SEMANTICALLY IDENTICAL TO A MACRO, THAT IS A SERIES OF EDIT COMMANDS EXECUTED WITH LATER RETURN TO THE COMMAND AFTER THE INVOCATION OF THE READ. THE NYU EDITOR SPECIFIES THAT ALL SAVED COMMANDS AT THE POINT OF THE READ ARE DESTROYED BEFORE BEGINNING THE READ. THE SETL VERSION ALLOWS RECURSIVE READS WITH THE CORRECT ITEMS SAVED, RETURNING CORRECTLY.

ANYTHING WHICH FULFILLS THE EDITORS REQUIREMENT TO BE A SYSTEM COMMAND IS NOT USED BY THE EDITOR AT ALL, BUT RATHER PASSED ONTO THE OPERATING SYSTEM. THIS GIVES THE EDITOR THE PROPERTY OF TRANSPARENCY. THIS MEANS THAT GIVEN AN OPERATING SYSTEM FOR TIMESHARING, THE EDITOR WONT AFFECT SEQUENCES OF CORRECT COMMANDS

TO THE OPERATING SYSTEM. HOWEVER, WITH THE ADDITION OF THE EDITOR, WE INTERPRET CERTAIN ERRONEOUS INPUTS TO THE OPERATING SYSTEM TO BE CORRECT INPUTS TO THE EDITOR. THIS PROPERTY OF TRANSPARANCY TO THE PREVIOUS LEVEL OF THE SYSTEM, IE, TO THE UNDERLYING OPERATING SYSTEM, MAKES AN EDITOR VERY VALUABLE. IT ADDS TO THE POWER OF THE OPERATING SYSTEM RATHER THAN TRYING TO BUILD ITS OWN POWER FROM SCRATCH. THE DISTINGUISHABLE FEATURE OF OPERATING COMMANDS (SYNTACTICALLY) FOR THE 6600 SERIES COMPUTERS IS THAT THEY ALL TERMINATE WITH EITHER A PERIOD OR A CLOSE PARENTHESES. THUS THE SYNTACTIC PRE-PROCESSOR TO THE EDITOR (WHICH MAY BE ANY AUTOMATIC SYNTAX ANALYZER) FORBIDS EDIT COMMANDS TO CONTAIN UNQUOTED PERIODS AND CLOSED PARENTHESES.

/* 3. PRELIMINARY CODE */

3.1 CHARACTER STRING UTILITIES

UTILITY ROUTINES HAVE A SPECIAL PLACE IN THE HEART OF SPECIFICATION PROGRAMMERS. EVEN MORE SO WHEN SETL IS THE SPECIFICATION LANGUAGE, BECAUSE IN SETL, THE ROUTINE IS INDEPENDENT NOT ONLY OF THE MACHINE AND OF THE OPERATING SYSTEM AND (SOMETIMES) OF OTHER ROUTINES, BUT ALSO INDEPENDENT OF THE DATA STRUCTURE EMPLOYED, MAKING COMPLICATED CHANGES IN THE STRUCTURE EXTREMELY SIMPLE.

A UTILITY ROUTINE IS DEFINED AS A ROUTINE (PRECEDED BY DOCUMENTATION AND FOLLOWED BY A TEST OF ITS PERFORMANCE) WITH THE FOLLOWING PROPERTIES:

1) IT IS SMALL. (THIS IS A PSYCHOLOGICAL NECESSITY.)

2) IT USES NO MACROS.

3) IT USES NO GLOBAL VARIABLES AND DOES NOT USE ANY VARIABLE TO TRANSMIT INFORMATION TO OTHER ROUTINES, EXCEPT ITS FORMAL PARAMETERS. THUS IT DOES NOT DEPEND ON INITIALIZATION PROCEDURES EXTERNAL TO ITSELF.

4) IT INITIALIZES ALL LOCAL VARIABLES BEFORE USING THEM.

5) IT IS COMPLETELY INDEPENDENT OF ALL OTHER ROUTINES AND DOES NOT CALL OTHER ROUTINES (EXCEPT OTHER UTILITIES OR SETL PRIMITIVES), OR DO ANY INPUT OR OUTPUT.

6) IT IS USEFUL IN AND OF ITSELF, NOT TAILOR MADE FOR THE PARTICULAR SITUATION USED FOR HERE.

7) IT HAS NO HOOKS INTO ANY PARTICULAR OPERATING SYSTEM.

THE PRELIMINARY CODE IS PRESENTED FIRST SINCE, IT EXEMPLIFIES THE DEPENDENCIES AND LACK OF DEPENDENCIES OF THE EDITOR SUBROUTINES. THE MACRO AND INITIALIZATION SECTIONS ARE PRESENTED NEAR THE BEGINNING SINCE MANY ROUTINES DEPEND ON THEM BUT THE UTILITIES ARE FIRST BECAUSE THEY HAVE NO DEPENDENCY ON THE MACROS OR THE GLOBAL VARIABLES INITIALIZED IN THE IMMEDIATELY FOLLOWING SECTIONS.

/* 2.2 POINTER VARIABLES IN SETL */

EVEN THOUGH SETL HAS NO POINTER TYPE VARIABLES PER SE, THE CURRENT SEMANTICS OF SETL ALLOW A SIMULATION OF FIRST LEVEL (NON-RECURSIVE) POINTERS WITH ONLY MODERATE INCONVENIENCE TO THE PROGRAMMER, AND N-LEVEL DEPTH POINTERS WITH SOME MORE INCONVENIENCE. TO SIMULATE A POINTER-TO-X TYPE VARIABLE WHERE X IS A VALID VARIABLE TYPE, SET UP A GLOBAL MAPPING CALLED POINTEDATBY WHOSE DOMAIN WE WILL DEFINE IN SUCH A WAY AS TO MAKE IT ≠POINT≠ TO ITS RANGE WHICH IS THE SET OF ALL DATA OBJECTS OF TYPE X TO WHICH YOU WISH TO HAVE IT POINT.

SO THE SET (MAPPING) POINTEDATBY CONTAINS EACH ORDERED PAIR WHOSE TAIL IS A NAME OF AN OBJECT OF TYPE-X THAT IS POINTED AT BY ITS HEA WHICH IS THE NAME OF THE OBJECT (DEFINED BELOW). MODIFICATIONS TO AN OBJECT WHOSE NAME IS POINTERTO(Y) ARE MADE BY ASSIGNMENTS TO POINTEDATBY(POINTERTO(Y)) RATHER THAN DIRECTLY TO Y, WHICH ACTUALLY HAS NO EXISTENCE OUTSIDE OF THE RANGE OF THE SET POINTEDATBY.

THE NAME OF DATA OBJECT X IS DEFINED AS A PAIR IN THE GLOBAL SET POINTERTO WHOSE HEAD IS A CHARACTER STRING BY WHICH THE PROGRAMMER TIME THE DATA OBJECT IS CREATED, THE FOLLOWING SEQUENCE OF ASSIGNMENTS MUST TAKE PLACE:

```
<≠Y≠,BLANK.> IN. POINTERTO;
POINTEDATBY(POINTERTO(≠Y≠)) = DATAOBJECT;
```

A MORE GENERAL ASSIGNMENT MAY BE MADE WITH THIS CODE:

```
IF POINTERTO(OBJECT) EQ. OM. THEN <OBJECT,BLANK.> IN. POINTERTO;;
POINTEDATBY(POINTERTO(≠Y≠)) = DATAOBJECT;
```

A MACRO ISPOINTER(X) CAN BE DEFINED TO BE THE BOOLEAN PREDICATE:

```
TYPE. X EQ. TUPL. AND. TYPE. HD. X EQ. STRING.
                  AND. TYPE. TL. X EQ. BLANK.
                  AND. POINTERTO(X) NE. OM.
                  AND. POINTEDATBY(POINTERTO(X)) NE. OM.
```

IT IS SURPRIZING HOW MANY OF THESE SUBROUTINES CAN BE CONSIDERED TO BE UTILITY ROUTINES EVEN BY SUCH A STRINGENT DEFINITION. HERE IS A LIST OF THE UTILITY ROUTINES DEFINED BELOW:

```
ABBR(KEY,TABLE)
ALN(STRING,AT)
BREAK (STRING,CHARSET)
DECODE(STRING)
FORMAT(LINE,FORMATS)
IHASH(STRING)
JHASH(STRING)
STRINGA LEXLT, STRING
NEXTTAB(COL,TABST)
A ORM, B
RIGHT(STRING,COL)
SPAN (STRING,CHARSET)
SQUEEZE(STRING)
CHARS, STRING
TRUNC, STRING
```

SOME NOTES ON CHARACTER STRINGS IN SETL

AT THE PRESENT TIME, THE SPECIFICATION OF SETL INDICATES THAT IF YOU ATTEMPT TO EXTRACT THE N+1 TH ELEMENT OF A CHARACTER STRING, YOU WILL RECEIVE THE VALUE OM, AND IF YOU ATTEMPT TO EXTRACT THE ZERO-TH ELEMENT OF A STRING, YOU WILL CRASH THE SYSTEM. FOR THE FOLLOWING STRING-MANIPULATION ALGORITHMS AND PERHAPS OTHER STRING ROUTINES AS WELL, IT WOULD ALLOW FOR LESS PROGRAMMING EFFORT AND A NICER LOOKING TEXT (THAT IS, SOURCE TEXT REQUIRING A SMALLER NUMBER OF CHARACTERS TO CONVEY ITS INTENT TO THE SETL COMPILER) IF BOTH OF THESE CASES WOULD RETURN THE VALUE NULC.

ABBR ATTEMPTS TO INTERPRET KEY AS AN ABBREVIATION (OR COPY) OF ONE OF THE ELEMENTS OF TABLE. SPECIFICALLY IT LOOKS FOR THE FIRST ELEMENT OF TABLE WHOSE FIRST CHARACTERS MATCH THOSE OF THE ABBREVIATION WITHOUT AMBIGUITY BETWEEN IT AND SUCCEEDING ELEMENTS. THE EDITOR (SEE THE MAIN LOOP) WILL INTERPRET AN AMBIGUOUS ABBREVIATION AS AN OPERATING SYSTEM COMMAND NOT AN EDIT COMMAND, UNLESS IT IS AN ABBREVIATION SPECIFICALLY APPEARING AS AN ENTRY IN THE TABLE.

```
DEFINEF ABBR(KEY,TABLE);
TABLE = TABLE + <<≠ ≠,0>>;
RETURN
IF ∃1<=K<=↓TABLE↑((TABLE(F)(1))(1:↓KEY) EQ.KEY AND.
   ((↓TABLE(K)(1)) EQ. ↓KEY OR. (TABLE(K+1)(1))(1:↓KEY) NE. KEY))
THEN TABLE(K)(2) ELSE OM.; END ABBR;
```

ALN JUSTIFIES A STRING LEFT, AFTER COLUMN COL.

```
DEFINEF ALN(STRING,COL);
IF COL GE. (↓STRING) OR. COL LE. 0 THEN RETURN STRING; END IF;
J = SPAN(STRING,S:≠ ≠>);
RETURN (COL*(≠ ≠))+STRING(J+1:↓STRING-J);
END ALN;
```

```
DEFINEF BLANKOUT(STRING); /* REMOVE ALL BLANKS IN STRING */
NEWSTRING = NULC.;
(∀1<=J<=↓STRING ↑ STRING(J) NE. ≠ ≠)
    NEWSTRING = NEWSTRING + STRING(J);;
```

```
RETURN NEWSTRING; END BLANKOUT;

                    BREAK RECOGNIZES A SPAN OF CHARACTERS AT THE
                    BEGINNING OF A STRING WHICH ARE NOT ELEMENTS OF
                    CHARSET, THE LEXICAL CLASS TO BE CHECKED.

DEFINEF BREAK(STRING,CHARSET); /* SNOBOL PATTERN--BREAK(CHARSET) */
J=1; (WHILE NOT.STRING(J)→CHARSET) J=J+1; END WHILE;
RETURN J-1; END BREAK;

DEFINEF DECODE(STRING);  /* UNDO SQUEEZE */
K=1;J=1;NEW=NULC.;
(WHILE STRING(K) NE. ZERO)
     NEW(J) = STRING(I) IS. PREV;  /* PLACE A CHARACTER */
     K = K+1; J = J+1;
     IF STRING(K) EQ. ZERO THEN  /* TAKE REPETITIONS */
          IF STRING(K+1) GT. 1 THEN
               NEW(J:CARD(K)) = CARD(K)*PREV;
               J = J+CARD(K);
               K = K+2;
          ELSE IF CARD(K+1) EQ. 1 THEN
               NEW(J) = ZERO;
               J = J+1;
               K = K+1;
          END IF STRING;
     END IF STRING;
     END WHILE;
RETURN NEW;END DECODE;

                    SIMULTANEOUSLY  TRANSLATE  CHARACTERS  USING  THE
                    MAPPING TABLE

DEFINEF TABLE ENCODE. STRING;
RETURN [+: 1<=K<=↓STRING] TABLE(STRING(K));
END TABLE ENCODE.;

DEFINEF TABLE DECODE. STRING;  /* UNDO ENCODE */
REVERSE = ≤ <TABLE(K,2),TABLE(K,1)>,1<=K<=↓TABLE ≥;
RETURN REVERSE ENCODE. STRING; END TABLE DECODE.;

   FORMAT REFORMATS A LINE BY RETURNING THE COLUMNS OF THE LINE IN
   THE ORDER SPECIFIED BY THE TABLE FORMATS.  EACH ENTRY IN FORMATS
   IS A PAIR INDICATING COLUMN COLUMNS, EG. <FROM,TO>.

FOR INSTANCE, IF TABLE = <<1,3>,<2,2>,<63,65>>
```

```
THEN FORMAT WILL RETURN THE CONCATENATION OF THE FOLLOWING COLUMNS:
                    1 2 3 2 63 64 65

DEFINEF FORMAT(STRING,FORMATS);RETURN
IF (↓FORMATS IS. LEN) LT. 1 THEN STRING
ELSE [↓;1<=K<=LEN] STRING(FORMATS(K)(1):FORMATS(K)(2));END FORMAT;
```

IHASH RETURNS THE HASHCODE OF A CHARACTER STRING BY TAKING EVERY 2 CHARACTERS AND ADDING THEIR DISPLAY CODE VALUE. THIS SUM IS THE NUMBER OF THE BIT SET IN THE BIT STRING WHICH IS RETURNED. OLDHASH IS THE HASH CODE OF SOME OTHER STRING TO WHICH THE THE CURRENT STRING IS BEING CONCATENATED FOR HASHING PURPOSES,WHICH COULD VERY WELL BE THE NULL STRING (NULL BIT STRING FOR THE HASH CODE). THE BIT STRING IS HASHED INTO A SETL SET FOR QUICK DETERMINATION OF MEMBERSHIP. IN A LOWER LEVEL LANGUAGE IMPLEMENTATION, OF COURSE, A BIT STRING IS PROBABLY FASTER THAN A RANDOM-ACCESS SET OF BITS.

```
DEFINEF IHASH(STRING,OLDHASH); PREV = INTEGER(STRING(1));
IF (↓STRING) GT. 1 THEN
   OLDHASH=OLDHASH+ ≤PREV+INTEGER(STRING(J))]S. PREV,2<=J<=↓STRING≥];
RETURN OLDHASH; END IHASH;
```

JHASH DOES THE SORE DIFFICULT (THAN IHASH) HIGHER LEVEL HASHING OF A STRING WITH DOUBLE COLONS INDICATING COLONS AND A SINGLE COLON BEING THE BREAK CHARACTER WHICH BRACKETS PATTERN DEFINITIONS WHICH ARE NOT HASHED AT ALL. FOR EACH BARE SECTION (NOT CONTAINING A PATTERN)OF THE STRING, IHASH IS CALLED TO ACCUMULATE THE HASHCODE.

```
DEFINEF JHASH(STRING);
HASH = NL.; COL = 1;
(WHILE COL LE. ↓STRING DOING COL = COL + 1;)
    HASHABLE = NULC.; COL = 1;
    (WHILE (STRING(J)EQ.(≠:≠) IMP. STRING(J+1)EQ.≠:≠)
         AND. COL LE. ↓STRING DOING COL = COL+1;)
        IF STRING(J) NE. ≠:≠ THEN HASHABLE = HASHABLE + STRING(COL);;
        END WHILE;
    IHASH=IHASH(HASHABLE,HASH); /* COMPUTE HASHCODE OF THIS PIECE */
    IF STRING(COL) EQ. ≠:≠ THEN /* SEARCH FOR A MATCHING COLON */
        COL = COL+1;J=COL;
        (WHILE STRING(COL) NE. ≠:≠ AND. COL LE. ↓STRING) COL=COL+1;;
        IF COL GE.↓STRING OR.(COL-J) GE.12 THEN ERRORS(4);RETURN F.;;
        END IF STRING;
```

```
    END WHILE COL;
RETURN HASH; END JHASH;
```

IT IS DEBATABLE WHETHER INTEGER, A MAPPING (COLLATING SEQUENCE) FROM THE CHARACTERS ONTO THE INTEGERS, CAN BE REFERRED TO IN A UTILITY ROUTINE, BUT IN GENERAL IT SHOULD BE A SETL PRIMITIVE RATHER THAN A GLOBAL VARIABLE, ALTHOUGH IN THE CURRENT IMPLEMENTATION IT IS EXACTLY THAT, A MACHINE DEPENDENT GLOBAL VARIABLE.

```
/* LEXLT DECIDES IF STRING A IS ≠LESS≠ THAN STRING B  */
DEFINEF STRINGA LEXLT, STRINGB;LEN=↓STRINGB MIN.↓STRINGA;RETURN
IF ∃1<=J<=LEN↑STRINGA(J) NE. STRINGB(J)
THEN INTEGER(STRINGA(J)) LT. INTEGER(STRINGB(J))
ELSE LEN EQ. ↓STRINGA; END STRINGA LEXLT. STRINGB;
```

GIVEN AN ORDERED SET OF TABS CALLED TABSET, FUNCTION NEXTTAB WILL RETURN THE NUMBER OF COLUMNS TO SKIP STARTING FROM COLUMN COL TO GET TO THE NEXT TAB AFTER COL.

```
DEFINEF NEXTTAB(COL,TABSET); RETURN
IF ∃1<=K<=↓TABSET↑(TABSET(K)IS.X) GE.COL THEN X-COL ELSE 1;
END NEXTTAB;
```

THIS USEFUL INFIX FUNCTION RETURNS ITS FIRST ARGUMENT AS ITS VALUE IF IT IS DEFINED, OTHERWISE THE SECOND ARGUMENTS.

```
DEFINEF A ORM. B; RETURN IF A NE. OM. THEN A ELSE B; END A ORM.;
```

THE FUNCTION RIGHT ALIGNS A STRING IT ON THE RIGHT MARGIN AND REDISTRIBUTES THE BLANKS EQUITABLY AMONG THE WORDS.

```
K=1;(∃K<=K<=↓STRING ↑ STRING(K)EQ.≠ ≠ AND.STRING(K+1)EQ.≠ ≠)
          STRING = STRING(1:K-1)+STRING(K+1:);;

DEFINEF RIGHT(STRING,MARGIN);
NEW=NULC.;BLVECTOR=NULT.;
/* NEW = STRING WITHOUT MULTIPLE BLANKS  */
NBLANKS = 0; NEW = NULC.;
(∀1<=COL<=MARGIN↑(STRING(COL)NE.(≠ ≠) OR. STRING(COL+1)NE.(≠ ≠)))
    NEW = NEW+STRING(COL); END ∀;
NBLANKS = [+:1<=K<=↓STRING ↑ STRING(K) EQ.  ≠ ≠] 1 ;
PRINT, NEW,COL,NBLANKS,↓STRING,↓NEW;
```

```
IF (↓STRING IS.LEN) LE. (MARGIN/2)OR.LEN GT.MARGIN THEN
   RETURN STRING;;

N = (MARGIN-↓NEW)/NRLANKS; /* N = NUMBER OF BLANKS PER WORD */
EXTRA = (MARGIN-↓NEW)/NBLANKS; /* NUMBER OF BLANKS LEFT OVER*/
BLVECTOR = [+:1<=K<=NBLANKS] <N>;
IF EXTRA GT. 0 THEN DIGITSET = ≤K,1<=K<=NBLANKS≥;
(∨1<=K<=EXTRA) X OUT. DIGITSET; BLVECTOR(X) = BLVECTOR(X)+1;;END IF;

/* SECOND PASS THROUGH NEW--PUT BLANKS BACK IN */
K=1; X = NULC.; PREV = 1;
(∨1<=COL<=(MARGIN)-1↑ NEW(COL)EQ.(≠ ≠))
    STRING = STRING + NEW(PREV+1:COL)+(BLVECTOR(K)*(≠ ≠)):
    PREV = COL; COL=COL+1; K=K+1; END ∨1;
RETURN STRING; END RIGHT;
```

SPAN IS A FUNCTION WHOSE ARGUMENT IS A SET. SPAN TRIES TO MATCH THE NEXT CHARACTER IN THE INPUT WITH ANY CHARACTER IN THE INDICATED SET. IF ITS NOT IN THE SET, SPAN FAILS. OTHERWISE, THE LIST IS SEARCHED FOR THE NEXT CHARACTER OF THE INPUT. IF IT IS NOT THERE, SPAN SUCCEEDS, BECAUSE AT LEAST ONE CHARACTER HAS BEEN MATCHED. IF THE SECOND ONE IS IS FOUND, THE THIRD IS TRIED, ETC.

EXAMPLE:

```
SET  =  ≤≠A≠,≠B≠,≠C≠,≠X≠, ≠Y≠, ≠Z≠≥

STRING = ≠ZABCCF≠

SPAN (STRING,SET) RETURNS 5 SINCE SET MATCHES
    THE FIRST 5 CHARACTERS OF STRING.
```

```
DEFINEF SPAN(STRING,CHARSET);  /* SNOBOL PATTERN */
J=1;(WHILE STRING(J)→CHARSET)J=J+1;END WHILE;
RETURN J-1;END SPAN;
```

FOR THE SAKE OF SMALLER, MORE COMPACT LIBRARIES, SQUEEZE CONDENSES OUT MULTIPLE OCURRANCES OF CHARACTERS. AS AN EXAMPLE IN OCTAL DISPLAY CODE 37 (45 OCTAL) BLANKS BECOMES 550043 INSTEAD OF 55555555...

```
DEFINEF SQUEEZE(STRING);
```

```
I=2;J=2;

(WHILE I LE. ↓STRING DOING I=I+1;J=J+1;)
     IF STRING(I+1) EQ. STRING(I) THEN N=2;;
(WHILE STRING(I+N) EQ. STRING(I) AND. N LE. 64) N = N+1; END WHILE;
IF N GE. 4 THEN STRING(J+1) = ZERO;
     STRING(J+2) = N-2;
     I=I+N+1;
     J=J+3;;
STRING(J) = STRING(I);
END WHILE;
RETURN STRING; END SQUEEZE;


          CHARS RETURNS THE SET OF CHARACTERS IN A STRING
          [+:X(J)→STRING]  ≤:X≥

DEFINEF CHARS. STRING;RETURN ≤STRING(J). 1<=J<=↓STRING≥;
END CHARS.;


                    TRUNC RETURNS A STRING WITHOUT ITS TRAILING
                    BLANKS.

DEFINEF TRUNC. STRING;
IF ∨ ↓STRING>=J>=1 ∧ STRING(J) EQ. (≠ ≠) THEN X=CM.;;
RETURN STRING(1:J); END TRUNC.;
```

```
            /*  3.2. MACRO SECTION       */

/* SYNTACTIC ABBREVIATIONS */

** CHECK(ITEM,TYP)=IF TYPE(ITEM) NE. TYP THEN ILLARG;**
** PUSH(ITEM) = STACK(↓STACK+1) = ITEM **
** POP(ITEM) = ITEM = STACK(↓STACK); STACK = STACK(1:↓STACK-1) **
** ILLARG = ERROR(1);OKFLAG=T.**          /* ILLEGAL ARGUMENT TYPE */
** ER = (≠$≠)**          /* USE $ FOR END-OF-RECORD */


/* ARGUMENT EXTRACTORS */

** ARG1 = ARG(1) **
** ARG2 = ARG(2) **
** ARG3 = ARG(3) **
** ARG4 = ARG(4) **



/* HASH-TABLE FIELD EXTRACTORS */

** PAGENO(X) = (X)(1) **
** LINENO(X) = (X)(2) **
** HASHCD(X) = (X)(3) **


/* FIELD EXTRACTORS OF SOFT PATTERNS */

** ELEN(X) =  X (1) **    /* MIN LENGTH */
** ETYP(X) =  X (2) **    /* 0 IS CHAR; 1 IS COLON */
** ESTR(X) =  X (3) **    /* TYPE 0--THE STRING;TYPE L--MULT STRING */
** EVAR(X) =  X (4) **    /* STORE IN VARIABLE */
** ELEX(X) =  X (5) **    /* LEXICAL CLASS TO BE MATCHED */
** EOPR(X) =  X (6) **    /* OPERATOR */
** EINT(X) =  X (7) **    /* OBJECT TO BE CREATED */
** ECNT(X) =  X (8) **    /* DYNAMIC COPY OF EINT */
** EQUICK(X)= X(↓X) **    /* SHORT FORM TO BE TRIED FIRST */
** EQROTHER(X) = (↓EQUICK(X)) GT. 0 **
```

```
            /*     3.3. INITIALIZATION    */

DEFINE INITIAL;

                        IT IS ASSUMED THAT ALL VARIABLES DEFINED IN THIS
                        SECTION ON INITIALIZATION ARE GLOBAL VARIABLES
                        WHICH ARE AUTOMATICALLY ATTACHED TO ANY SUBROUTINE
                        USING THEM. ALL THE REST OF THE VARIABLES IN THE
                        PROGRAM ARE LOCAL, AND HENCE ARE INITIALIZED
                        BEFORE USING A THEM IN A SUBROUTINE.


          /* LEXICAL CLASS INITIAL DEFINITIONS */


/* THE INITIAL LEXICAL CLASSES ARE DEFINED AS FOLLOWS:

          A   ALPHABETIC CHARACTERS
          N   NUMERIC CHARACTERS (0-9)
          S   SPECIAL CHARACTERS (ANYTHING ELSE)
          O   ARITHMETIC OPERATORS (+ - * /)
          B   BLANK
              OTHERS ARE ALL INITIALLY NULL                    */

ALPH = ≠ABCDEFGHIJKLMNOPQRSTUVWXYZ≠; NUM = ≠1234567890≠;

/* CHARS IS A UTILITY WHICH CHANGES A STRING
    INTO THE SET OF ITS CHARACTERS  */

NUMS = CHARS. NUM; ALPHS = CHARS. ALPH;
ALPHANUM = ALPHS + NUMS;
QUOTESET = CHARS. ≠↓≤≥^↑/=¬][<>$Ξ→≠;
SPECIALS = CHARS. (≠.,:;()~≠) + QUOTESET LESS. ≠/≠;
OP = ≠+*-/≠; OPS = CHARS. OP;

/* LXCLASS IS THE SET OF LEXICAL CLASSES */
LXCLASS = ≤;<≠A≠,ALPHS>,<≠N≠,NUMS>,<≠O≠,OPS>,<≠B≠,≤:≠ ≠≥>,
          <≠S≠,SPECIALS>≥;

 /* ANSOB IS THE SET OF CHARACTERS ≠ALSO≠ */
ANSOB = CHARS. ([+:X→LXCLASS] HD. X);

/* SCALER IS THE MASTER SCALE STRING */
SCALER = [+: 1<= J<=15]
         ([+: 1<= K<=9] DEC.K + ALPH(J));

/* SYSGUESS IS THE TUPLE OF PATTERNS FOR ROUTINE GUESS TO DETERMINE
THE SYSTEM FROM LINE 1 OF THE FILE.    */
```

```
SYSGUESS=<≠ PROGRAM: ::(A)>1:≠,
          ≠ OVERLAY: ::(N)>1:≠,
          ≠ SUBROUTINE: ::(A)>1:≠,
          ≠ FUNCTION: ::(A)>1:≠,
          ≠ IDENT: ::(A)>1:≠,
          ≠ ID: :DIVISION≠,
          ≠/*≠,
          ≠BEGIN≠,
          ≠$≠,
          ≠DEFINE≠,
          ≠Ξ≠,
          ≠!≠,
          ≠:↑1:*:(A)=1:≠,
          ≠ ≠>;

/* ≠WHO≠ IS THE TUPLE OF SYSTEM NAMES LISTED IN SYSGUESS.      */
WHO = <   ≠FORTRAN≠,≠FORTRAN≠,≠FORTRAN≠,≠FORTRAN≠,
          ≠COMPASS≠,
          ≠COBOL≠,
          ≠PL1≠, ≠PL1≠,
          ≠ALGOL≠,
          ≠SETLB≠,≠SETLB≠,
          ≠SYMPL≠,
          ≠BALM≠,
          ≠UPDATE≠,
          ≠ABSOLUTE≠>;

INTERPS = ≤: ≠FORTRAN≠,≠BASIC≠,≠SIMSCRIPT≠,≠ALGOL≠,≠UPDATE≠>;

SYSTABS = ≤:
          <≠ABSOLUTE≠>,
          <≠ALGOL≠,     7,10,13,16,19>,
          <≠BAL≠,       10,16,31,42,72>,
          <≠BALM≠,      7,11,15,19,23,27,31,35>,
          <≠COBOL≠,     8,12,16,20,24>,
          <≠COMPASS≠,   11,21,35>,
          <≠BAS≠,       8,12,36,35,40,45,50,55,60,65>,
          <≠JOVIAL≠,    8,18,28,50,60>,
          <≠MIX≠,       12,17,22,30>,
          <≠PL1≠,       12,15,18,21,24,27,30>,
          <≠QUIDDL≠,    8,12,16,20,24>,
          <≠SETLB≠,     7,11,15,19,23,27,31,35>,
          <≠SIMSCRIPT≠>,
          <≠FORTRAN≠>,
          <≠SIMULA≠,    7,10,13,16,19>,
          <≠SNOBOL≠>,
          <≠SYMPL≠,     8,18,28,50,60>,
          <≠T1≠>,
          <≠T2≠> ≥;
```

THIS TABLE CONSISTS OF AN ORDERED COLLECTION OF PAIRS WHOSE HEADS ARE CHARACTER STRINGS GIVING THE NAME OF EDIT COMMANDS AND WHOSE CORRESPOINDING TAILS ARE A COPY OF THE CODE USED TO IMPLEMENT THE EDIT COMMAND. SEE THE CHAPTER CALLED FIRST LEVEL EDIT COMMANDS. IT IS A DEFINITE DEFICIENCY IN SETL NOT TO HAVE VARIABLES OF TYPE POINTER. (SEE SEC 2.3)

```
EXECUTES = <
          <≠A≠          , ANCHOR>,
          <≠ALIGN≠      , ALN>,
          <≠ANCHOR≠     , ANCHOR>,
          <≠ASK≠        , ASK>,
          <≠B≠          , BOTTOM>,
          <≠BLANK≠      , ZBLANK>,
          <≠C≠          , CHANGE>,
          <≠DELETE≠     , DELETE>,
          <≠DISPLAY≠    , DISPLAY>,
          <≠DUPLICATE≠, DUPL>,
          <≠DYNAMIC≠    , DYNAMIC>,
          <≠ECHO≠       , ECHO>,
          <≠EDIT≠       , ZEDIT>,
          <≠END≠        , ENDEDIT>,
          <≠ERROR≠      , ZERROR>,
          <≠ESCAPE≠     , ESCAPE>,
          <≠EXECUTE≠    , ZEXECUTE>,
          <≠FORMAT≠     , FORMAT>,
          <≠HALTREAD≠   , HALTREAD>,
          <≠I≠          , INSERT>,
          <≠IFEQ≠       , IFEQ>,
          <≠IFGE≠       , IFGE>,
          <≠IFGT≠       , IFGT>,
          <≠IFLE≠       , IFLE>,
          <≠IFLT≠       , IFLT>,
          <≠IFNE≠       , IFNE>,
          <≠IFTYPE≠     , IFTYPE>,
          <≠INSERT≠     , INSERT>,
          <≠L≠          , LOCATE>,
          <≠LIST≠       , LIST>,
          <≠LOCATE≠     , LOCATE>,
          <≠LZONE≠      , LZONE>,
          <≠M≠          , MASTER>,
          <≠LEXCLASS≠   , LEXCLASS>,
          <≠MACRO≠      , MACRO>,
          <≠MARGIN≠     , MARGIN>,
          <≠MASTER≠     , MASTER>,
          <≠MOVE≠       , MOVE>,
          <≠MSTOP≠      , MSTOP>,
          <≠MZONE≠      , MZONE>,
```

```
        <≠N≠          , NEXT>,
        <≠NEXT≠       , NEXT>,
        <≠NOCC≠       , NOCC>,
        <≠NODYNAMIC≠  , NODYNAMIC>,
        <≠NOECHO≠     , NOECHO>,
        <≠NOEXECUTE≠  , NOEXECUTE>,
        <≠NOLIST≠     , NOLIST>,
        <≠NONUMBER≠   , NONUMBER>,
        <≠NOPAGE≠     , NOPAGE>,
        <≠NOPROMPT≠   , NOPROMPT>,
        <≠NOUPD≠      , NOUPD>,
        <≠NUMBER≠     , NUMBER>,
        <≠OVERLAY≠    , OVERLAY>,
        <≠P≠          , PRINT>,
        <≠PAGE≠       , PAGE>,
        <≠PRINT≠      , PRINT>,
        <≠PROMPT≠     , PROMPT>,
        <≠PZONE≠      , PZONE>,
        <≠R≠          , REPLACE>,
        <≠READ≠       , READ>,
        <≠REMOVE≠     , REMOVE>,
        <≠REPLACE≠    , REPLACE>,
        <≠RESUME≠     , RESUME>,
        <≠SAVE≠       , SAVE>,
        <≠SAFETY≠     , SAFETY>,
        <≠SCALE≠      , SCALE>,
        <≠SETVAR≠     , SETVAR>,
        <≠SHOW≠       , SHOW>,
        <≠SKIP≠       , SKIP>,
        <≠SPLIT≠      , SPLIT>,
        <≠STATUS≠     , STATUS>,
        <≠SUSPEND≠    , SUSPEND>,
        <≠SYSTEM≠     , SYSTEM>,
        <≠T≠          , TOP>,
        <≠TABS≠       , SYSTEM>,
        <≠TIME≠       , TIMER>,
        <≠TOP≠        , TOP>,
        <≠UNSAFE≠     , UNSAFE>,
        <≠UPD≠        , UPD>,
        <≠ZONE≠       , ZONE>   >;

DEFAULT                         VALUES OF FLAGS AND PARAMETERS

ALTREAD = F.;                   /* WE ARE NOT READING AN ALTERNATE FILE */
ANCHORQ = F.;                   /* LOCATE ANYWHERE */
ARG = NULL.;                    /* LEXICAL OUTPUT--VECTOR OF ARGS */
CCING = F.;                     /* NO CARRIAGE CONTROL MODE */
```

```
CHBLANK = ≠$≠;                  /* BLANK CHARACTER (SEE OVERLAY) */
CHCONCAT = ≠↓≠;                 /* INTRA MACRO CONCATENTATION CHARACTER */
CHESCAPE = ≠$≠;                 /* ESCAPE CHARACTER  */
CHTAB = ≠↑≠;                    /* TAB CHARACTER */
CS = NULC.;                     /* LAST CHANGE STRING */
DELFLAG = F.;                   /* WE ARE NOT IN DELETE MODE */
DEVICE = 72 ;                   /* DEVICE WIDTH (IGNORED) */
DYNAMING = F.;                  /* DO NOT SAVE DIRECTIVES */
EOFYET = F.;                    /* WE HAVE NOT READ AN EOF */
ECHOING = T.;                   /* ECHO CHANGES */
FNT = NL.;                      /* NO FILES IN FNT TO START */
HOWSAFE = 25;                   /* SAFETY LIMIT */
INITFLAG = T.;                  /* ARE WE EDITING A NEW FILE */
INFIT = F.;                     /* NOT IN AN INFINITE LOOP */
ITSSAFE = T.;                   /* PROTECTED MODES */
LISTFILE = OM.;                 /* PAGED OUTPUT */
LISTSET = ≤:≠F≠≥;               /* LIST OPTION  E */
LP = 0;                         /* LINE POINTER */
LPLINES = 50;                   /* LINE PRINTER LINES PER PAGE */
LPSAVE = 0;                     /* SAVED LP */
LS = NULC.;                     /* LAST LOCATE STRING */
LZONE1 = 1;                     /* LOCATE - CHANGE ZONE */
LZONE2 = 72;
MACRING = F.;                   /* WE ARE NOT EXPANDING A MACRO */
MACROTBL = NULT.;               /* MACRO TABLE */
MAXPAGES = 4;                   /* PAGES OF VIRTUAL MEMORY ALLOWED */
MS = ≠*       ≠;                /* MASTER STRING */
MZONE1 = 11;                    /* MASTER ZONE */
MZONE2 = 72;
NOINFIT = F.;                   /* INFINITE LOOPS NOT SHUT OUT */
NCHANGES = 0;                   /* NUMBER OF CHANGES SINCE LAST SAVE */
NUMBERQ = T.;                   /* NUMBER DISPLAYED LINES */
PLUS = F.;                      /* LOCATING FORWARD OR BACKWARD */
PROMPTQ = T.;                   /* PROMPT DURING INSERT MODE */
PAGING = F.;                    /* WE ARE NOT PAGING */
POWERON = T.;                   /* MONITOR MAIN LOOP EMPOWERED */
PZONE1 = 1;                     /* PRINT ZONE */
PZONE2 = 72;
SKIPPER = NULC.;                /* O.S. SKIPPING STRING */
SKIPSTOP  = ≠HELP≠;             /* UNIVERSAL SKIP STOP LABEL */
SYSTNAME = ≠ABSOLUTE≠;          /* SYSTEM NAME */
TABLE = NULT.;                  /* HASH TABLE */
TABSET = NL.;                   /* SET OF TABS FOR THIS SYSTEM */
TTYNAME = ≠INPUT≠;
TTYOUT = ≠OUTPUT≠;
UPDING = F.;                    /* NOT UPDATE TEXT */
RETURN; END INITIAL;
```

/* 3.4 INPUT OUTPUT SIMULATION */

SINCE SETL INPUT/OUTPUT HAS NOT YET BEEN COMPLETLY DEFINED, I HAVE SPECIFIED THE FOLLOWING ROUTINES TO ACCOMODATE THE NEEDS OF THE EDITOR. (THIS SECTION WILL BE SUPERSEDED BY THE SOON-TO-BE PUBLISHED THESIS OF GEORGE WINEBERG. BOTH SEQUENTIAL AND RANDOM FILES CAN BE SIMULATED USING THE SETL CAPABILITY OF HANDLING AN UNLIMITED NUMBER OF EXTERNAL FILES. THIS ELIMINATES THE UNWIELDY LIMITATION THAT ALL RANDOM FILES AND ALL OUTPUT FILES EXCEPT TWO NAMELY OUTPUT AND OUTFILE MUST BE SIMULATED BY SETS HELD IN MEMORY.

AN OPERATING SYSTEM IS A TUPLE CONSISTING OF TWO COMPONENTS:

OPERATING SYSTEM = <FILES, CONTROL-POINTS>

A FILE IS A PHYSICAL ENTITY -- SAY THE BITS ON A FEW INCHES OF MAGNETIC TAPE, WHILE A CONTROL POINT IS A SOFTWARE DATA OBJECT WHICH CORRESPONDS TO A SINGLE RESTARTABLE COMPUTER PROCESS.

SETL JOBS IN THE SCOPE OPERATING SYSTEM ARE USUALLY CONSIDERED FROM THE VIEWPOINT OF A SINGLE CONTROL POINT. A NOTABLE EXCEPTION IS PETER MARKSTEINS OPERATING SYSTEM IN SETL WHICH IS DEPENDENT ON HAVING MANY CONTROL POINTS AVAILABLE FOR PARALLEL COMPUTATIONS. HOWEVER, ANY SETL JOB IS PERMITTED TO OPEN AN INDEFINITE NUMBER OF FILES. ALL EXTERNAL INFORMATION USED BY A JOB AT A CONTROL POINT IS EITHER A FILE OR A PART OF A FILE.

THE FILE NAME TABLE (FNT) CONSISTS OF A SET OF POINTERS TO FILES. SO EACH ELEMENT OF FNT LOOKS LIKE THIS:

<FILENAME,FILE>

THE FILE NAME MUST ARBITRARILY SATISFY THE REQUIREMENTS STATED IN THE BOOLEAN PREDICATE VALID. THIS APPEARS IN THE EDIT SUBROUTINE SECTION. IT MUST BE SEVEN OR FEWER ALPHABETIC CHARACTERS BEGINNING WITH AN ALPHABETIC.

FILE = <TAPEHEAD, RECORDS>

A FILE CONSISTS OF A TAPEHEAD AND A VARIABLE LENGTH VECTOR OF RECORDS. THE TAPEHEAD POINTS TO THE RECORD WE ARE ABOUT TO READ. REWIND REWINDS A FILE BY SETTING THE TAPEHEAD OF THE FILE TO ONE. IF WE HAVE READ THE LAST RECORD, TAPEHEAD POINTS TO THE NON-EXISTANT (#RECORDS+1) RECORD AND ALL FURTHER READS RETURN THE END-OF-FILE MARK WHICH IS REPRESENTED IN SETL BY OM. OM HAPPENS TO BE THE INTERNAL EQUIVALENT OF A PHYSICAL END-OF-FILE MARK WHICH IS

REPRESENTED ON A TAPE BY AN END-OF-RECORD MARK (DESCRIBED BELOW) FOLLOWED BY THE CHARACTER OM., FOLLOWED BY FIVE AND A HALF INCHES OF BLANK TAPE, OR THE EQUIVALENT ON A DISK. NOTE THAT ALL DISK FILES ARE ACCESSED AS IF THEY WERE TAPE FILES. EVEN WHEN INTERPRETING THE INFORMATION ON A ROTATING MASS STORAGE DEVICE (DISK OR DRUM) AS A RANDOM FILE (DESCRIBED BELOW), THE READ AND WRITE COMMANDS ACT AS IF EACH RANDOM RECORD WERE A TAPE FILE OPENED FOR SEQUENTIAL ACCESS. THE OPERATING SYSTEM SOFTWARE IS EXPECTED TO TAKE CARE OF THIS EXTREMELY CONVENIENT ASSUMPTION. THE SIMULATED MINI-OPERATING SYSTEM GIVEN HERE DOES TAKE CARE OF IT, THOUGH INEFFICIENTLY.

RECORD = <LINE1,LINE2,...,ER>

A RECORD IS AN ARTIFICIAL ENTITY OF VARIABLE LENGTH (ALTHOUGH OCCASIONALLY ASSIGNED A FIXED LENGTH BY THE USER). IT IS A TUPLE OF ITEMS FOLLOWED BY AN END-OF-RECORD MARK, REPRESENTED IN MEMORY BY THE SETL CHARACTER EP, WHICH IS UNIQUE ONLY IN THE FACT THAT IT IS A CHARACTER USED ONLY FOR INTERNALLY REPRESENTING END-OF-RECORD MARKS, WHICH APPEAR EXTERNALLY ON A TAPE AS ONE HALF INCH OF SKIPPED TAPE FOLLOWED BY THE END OF RECORD CHARACTER.

THE NEXT PARAGRAPH PERTAINS ONLY TO THE IMPLEMENTED SYSTEM AT NYU AND CAN BE DISREGARDED BY THE CASUAL READER OR THE NON-NYU USER. A FILE IS A BALM VECTOR CONSISTING OF A LOCAL FILE NAME, A BUFFER LENGTH, AND TWO BUFFERS. I HOPE THAT A FILE CAN BE AN ELEMENT OF A SET. IN THE FOLLOWING ROUTINES THE ARGUMENT ≠FILE≠ IS A SETL CHARACTER STRING AND ≠BFILE≠ IS A NAME WHOSE VALUE IS A POINTER-TO-A-BALM-VECTOR, WHICH ≠IS≠ BY DEFINITION, A SETL FILE. THE FILE IS CREATED AND OPENED FOR SEQUENTIAL ACCESS AND REWOUND. RECLENGTH SHOULD BE FROM 2 TO 132 CHARACTERS LONG. FILE SHOULD BE A VALID FILE NAME, THAT IS, A CHARACTER STRING CONSISTING OF AN ALPHABETIC FOLLOWED BY UP TO 6 NUMERICS, OR AN INTEGER N WHICH IS AN ABBREVIATION FOR THE LOCAL FILE NAME ≠TAPEN≠.

THE UNDERLYING BALMSETL PRIMITIVES AND PROCEDURES ARE:

```
BFILE = MAKFILE(FILE, RECLENGTH)  CREATE FILE
REWIND(BFILE)                     REWIND FILE
SKIP(BFILE, N)                    SKIP N ITEMS ON BFILE
WRITSETL(BFILE, ITEM)             WRITE ITEM ON BFILE
X = READSETL(BFILE)               READ NEXT ITEM FROM BFILE
```

THE SEQUENTIAL FILE COMMANDS ARE:

```
CREATE(FILE)                CREATE A FILE IF ITS NOT ALREADY THERE
EVICT(FILE)                 EVICT A FILE (RANDOM OR SEQUENTIAL)
REWINDF(FILE)               REWIND A FILE (RANDOM OR SEQUENTIAL)
COPYF(FILE1, FILE2)         COPY FILE1 TO FILE2
SKIPF(FILE, NREC)           SKIP NREC ITEMS ON FILE
WRITEF(FILE, STRING)        WRITE STRING ON FILE
```

```
X=READF(FILE)                    X IS NEXT ITEM ON FILE
```

THE RANDOM FILE COMMANDS ARE:

```
OPENR(FILE, LEN)                 LEN = LENGTH OF THE DIRECTORY
ITEM =READR(FILE, RECORD)        ITEM IS A COPY OF RECORD IN FILE
WRITER(FILE, RECORD, ITEM)       WRITE ITEM AS RECORD OF FILE
```

TO USE A RANDOM FILE IN SETL, YOU MAY REFER TO UP TO LEN RECORDS AS SPECIFIED ON THE OPENR CARD. JUST ASUME THAT AS RECORD IS THERE AND USE WRITER TO WRITE ON IT. DO NOT TRY TO WRITE ON MORE THAN LEN RECORDS.

FUNCTION VALID RETURNS TRUE IF ITS ARGUMENT IS A STRING WHICH SATISFIES THE REQUIREMENTS TO BE A VALID FILE NAME.

```
DEFINEF VALID. FILE;RETURN FILE(1)→ALPHS AND. (↓FILE) LE. 7;
END VALID.;


DEFINE COPYB(BFILEA,BFILEB); /* COPY ONE BALM FILE TO ANOTHER */
X=OM.;
(WHILE NOT. X EQ. EB)
    X=READSETL(BFILEA);
    WRITSETL(BFILEB,X);
    END WHILE;
RETURN; END COPYB;


DEFINE COPYF(FILEA,FILEB);COPYBF(FNT(FILEA),FNT(FILEB));
RETURN; END COPYF;


DEFINE CREATE(FILE);
IF FNT(FILE) EQ. OM. THEN FNT(FILE) = MAKFILE(FILE,132);;
RETURN; END CREATE;


DEFINE EVICT(FILE);FNT(FILE)=OM.;RANINDEX(FILE)=OM.;RETURN;END EVICT;


DEFINE OPENR(FILE,LEN);   /* OPEN RANDOM FILE */
/* LEN IS THE LENGTH OF THE DIRECTORY */
IF NOT. VALID. FILE OR. LEN LE. 1
THEN PRINT. ≠ILLEGAL ARG≠,FILE,LEN;RETURN;;
IF (FNT(FILE) IS. BFILE) NE. OM. THEN
    REWIND(BFILE); RANINDEX(FILE) = READSETL(BFILE);
ELSE RANINDEX(FILE) = NULT.; FNT(FILE) = ≠RANDOM≠; END IF;
```

```
RETURN; END OPENR;

DEFINEF READF(FILE); RETURN READSETL(FNT(FILE)); END READF;

DEFINEF READR(FILE,RECORD);
IF FNT(FILE) NE. ≠RANDOM≠ THEN PRINT.≠BAD READ RANDOM≠;RETURN OM.;;
I = RANINDEX(FILE,RECORD); IF I EQ. OM. THEN RETURN OM.;;
RETURN READF(I); END READR;

DEFINE REWINDF(FILE); CREATF(FILE);
REWIND(FNT(FILE));RETURN;END REWINDF;

DEFINE SKIP(BFILE,NREC);
(~1<=K<=NREC)Z=READSETL(BFILE); END ~1;Z=OM.;RETURN; END SKIP;

DEFINE SKIPF(FILE,NREC);SKIP(FNT(FILE),NREC);RETURN;END SKIPF;

DEFINE WRITEF(FILE,STRING); CREATE(FILE);
WRITSETL(FNT(FILE),STRING); RETURN; END WRITEF;

DEFINE WRITER(FILE,RECORD,ITEM);
IF(RANINDEX(FILE) IS.X) EQ. OM. THEN OPENR(FILE);X = RANINDEX(FILE);;
IF NOT. RECORD + X THEN RANINDEX(FILE) = X+<RECORD>;;
CREATE((FILE+RECORD+(≠Ξ≠))IS.J);
WRITEF(J,ITEM); RETURN; END WRITER;
```

/* 4.1 INSERTING */

THE ROUTINES IN THIS SECTION ARE USED WHEN A NEW LINE IS TO BE INSERTED INTO THE FILE. DOINSERT IS THE BASIC ROUTINE WHICH CONTROLS INSERTION. THE INSERT, OVERLAY, AND MASTER COMMANDS ALL INVOKE DOINSERT.

```
DEFINE DOINSERT;
LINE =NULC.;
IF NOT. PROMPTQ THEN PRINT. ≠ENTER LINES≠; END IF NOT.;
(WHILE LINE(1:2) NE. ≠↵ ≠)
    IF PROMPTQ THEN PRINT. DEC. LP + ≠>≠; END IF PROMPTQ;
    READ. LINE;
    IF LINE(1) EQ. CHESCAPE THEN X = LINE(2);
        IF X EQ. CHESCAPE THEN LINE = LINE(2:↓LINE); ELSE
        IF X → NUMS THEN PUTEOR. DEC. X; ELSE
        IF X → ALPHS THEN PUTFILE. LINE(2:7);;
    ELSE PUT(IMODE(LINE)); LP = LP + 1; END IF LINE;
    END WHILE;
RETURN; END DOINSERT;


DEFINE DOOVRLAY(STRING);
J=SPAN(X,CHARS(≠$≠));
IF J GT. 0 THEN X(1:J) = J * (≠ ≠);;
J=SPAN(X,CHARS(≠ ≠));
IF J GT. 0 THEN STRING = STRING(J+1:↓STRING-J); ELSE J=0; END IF;
X = GET(LP);
X(LZONE1+J:(↓X) MIN.(LZONE2+1-LZONE2)) = STRING;
PUT(X); RETURN; END DOOVRLAY;


DEFINEF DOTABS(STRING,TABSET); /* EXPAND THE TABS IN STRING */
/* NO TAB FOR SYS ARS */
IF ↓TABSET EQ. 0 THEN RETURN STRING; END IF;
WITHTABS =NULC.;
STRING = SYSINTRP(SYSTNAME,STRING);
(↓1<=K<=↓STRING)
    X = STRING(K);
    IF X NE. CHTAB THEN WITHTABS = WITHTABS + X;
    ELSE WITHTABS = WITHTABS + NEXTTAB(K,TABSET) * (≠ ≠);
    END IF;
    END ↓;
RETURN WITHTABS; END DOTABS;
```

IMODE PREPROCESSES THE STRING CALLING DOTABS TO PROCESS THE TAB CHARACTERS AND SYSINTRP TO PROCESS A POSSIBLE INTERPRETIVE SYSTEM. THE LOWER LEVEL ROUTINE PUT (SEE SECTION 4.4) IS FINALLY CALLED TO PUT THE PROCESSED LINE INTO THE FILE AT LOCATION LP,MOVING ALL THE OTHER LINES DOWN ONE.

```
DEFINEF IMODE(STRING);   /* UNRAVEL THE INSERTION STRING */
WITHTABS = DOTABS(STRING,SYSTABS(SYSTNAME));
NEW = NULC.;
(∨1<=K<=↓WITHTABS)
    NEW(K)= IF WITHTABS(K) EQ. ≠ ≠ THEN MASTLINE(K) ELSE WITHTABS(K);
    END ∨;
WITHTABS = NULC.; X = TRUNC. NEW(1:LZONE2 MIN. ↓STRING); PRINT. X;
RETURN X;
END IMODE;


DEFINEF SYSALGOL(STRING);
I=1;(WHILE STRING(I) EQ. ≠ ≠) I=I+1; END WHILE;
STRING = STRING(I:↓STRING); RETURN
IF STRING(1)→ALPHS THEN (10*(≠*≠)) + STRING
ELSE IF STRING(1) EQ. ≠/≠ THEN (5*(≠*≠))+STRING
ELSE STRING;
END SYSALGOL;


DEFINEF SYSBASIC(STRING);
/* ENFORCE THE LINE NUMBER CONVENTION AND TURN OVER TO SYS FTN. */
X=SPAN(NUMS WITH. ≠ ≠);
RETURN IF X EQ. 0 THEN OM. ELSE
   SYSFTN(BLANKOUT(STRING(1:X)))+STRING(X+1:↓STRING-X);
END SYSBASIC;
```

FUNCTION SYSFTN DOES A TRANSFORMATION ON A STRING CORRESPONDING TO SYSTEM FORTRAN,NAMELY ALPHABETICS START IN COL.7, NUMERICS BECOME LABELS NICELY FORMATTED, A PLUS FOR CONTINUATION AND A STAR FOR COMMENTS. A LINE BEGINNING WITH THE ESCAPE CHARACTER IS RETURNED UNTOUCHED. ANYTHING ELSE IS AN ERROR.

```
DEFINEF SYSFTN(STRING);
    J=1;(WHILE STRING(J) EQ. (≠ ≠)) J=J+1; END WHILE;
    STRING = STRING(J:↓STRING); X=STRING(1); RETURN
```

```
    IF X → ALPHS OR
         X EQ. ≠C≠ AND. STRING(2) → (ALPHANUMS+OPS WITH. ≠=≠)
         THEN (6*(≠ ≠)) + STRING
    ELSE IF X EQ.≠*≠ OR. X EQ. CHESCAPE THEN  STRING
    ELSE IF X→ NUMS AND. E2<=J<=5 ↑ NOT. STRING(J)→NUMS THEN
        ((5-J)*(≠ ≠)) + STRING(1:J) + (≠ ≠) + STRING(J+1:↓STRING)
    ELSE IF X EQ. ≠+≠ THEN  (5*(≠ ≠)) + STRING ELSE OM.;
END SYSFTN;
```

UPDATE, REQUIRES TEXT INTERSPERSED WITH DIRECTIVES WHOSE FORMAT IS A STAR IN COLUMN ONE FOLLOWED BY VARIOUS ARGUMENTS. THIS IS NOT INCLUDED BECAUSE OF ITS INTRINSIC VALUE OR INTEREST BUT ONLY AS AN EXAMPLE OF HOW TO IMPLEMENT AN INTERPRETIVE SYSTEM WITHIN THE EDITOR. IT HAS THE EFFECT OF EXPANDING THE ABBREVIATIONS FOR FOUR OF THE MOST COMMON UPDATE DIRECTIVES. REMEMBER ANY ERROR IN AN INTERPRETIVE SYSTEM LINE EXPANSION MUST ABORT THE USER OUT OF THE CURRENT LINE TO PREVENT UNDESIREABLE ITERATIONS AND AFTER-EFFECTS.

```
DEFINEF SYSUPDATE(STRING);
I=1;(WHILE STRING(K) EQ. ≠ ≠) I=I+1; END WHILE;
IF STRING(I) NE. ≠*≠ THEN RETURN STRING;;
X=STRING(I+1:2);Z = STRING(I+4:↓STRING);
RETURN   IF X EQ. ≠I ≠ THEN ≠*INSERT ≠+Z
    ELSE IF X EQ. ≠D ≠ THEN ≠*DELETE ≠+Z
    ELSE IF X EQ. ≠B ≠ THEN ≠*BEFORE ≠ +Z
    ELSE IF X EQ. ≠C ≠ THEN ≠*COMPILE ≠+Z
    ELSE X + Z;
END SYSUPDATE;


DEFINEF SYSINTRP(SYSNAME,STRING); RETURN
IF SYSNAME → INTERPS THEN
    IF SYSNAME EQ. ≠FORTRAN≠ THEN SYSFTN(STRING)
    ELSE IF SYSNAME EQ. ≠ALGOL≠ THEN SYSALGOL(STRING)
    ELSE IF SYSNAME EQ. ≠BASIC≠ THEN SYSBASIC(STRING)
    ELSE IF SYSNAME EQ. ≠SIMSCRIPT≠ THEN SYSFTN(STRING)
    ELSE SYSUPDATE(STRING)
ELSE STRING;
END SYSINTRP;
```

/* 4.2 LOCATE PROCESS */

FIRST LEVEL ROUTINES ARE INVOKED BY THE MASTER INTERPRETER (EDIT LOOP). THEY ARE LOCATE AND CHANGE.

LOCATE ARG1 ARG4
CHANGE ARG1 ARG2 ARG3 ARG4

ARG1 IS THE LOCATE STRING (LS) TO BE MATCHED AGAINST THE LINES OF THE FILE. ARG2 IS THE CHANGE STRING SO THAT IF THE LS IS FOUND IT WILL BE ELIMINATED AND REPLACED BY THE CS. ARG3 IS THE REPETITION FACTOR--HOW MANY TIMES MUST THE LS BE CHANGED TO THE CS IN THE LINE IN WHICH THE LS IS FIRST FOUND. IF ARG4 IS NOT NULL, THEN CREATE AN INFINITE LOOP, EFFECTIVELY MATCHING ALL OCCURANCES OF LS FROM THE CURRENT POSITION OF THE LINE POINTER (LP) TO THE LOCATION OF THE LINE WHICH MATCHES ARG4.

LOCATE IMMEDIATELY [illegible] LOOP. AT THE NEXT LEVEL, LOCATE LIMIT, [illegible] OCCURANCES OF LS ON A LINE SUCH THAT THE ITERATION FORWARD OR BACKWARDS NEVER [illegible] PASSES LIMIT DURING THE LOCATE [illegible]

[illegible] STRINGS, EACH CONSISTING OF EITHER A COLON DELIMITED PATTERN OR AN UN[illegible] BECAUSE THE QUOTES [illegible] OF [illegible] OF CHARACTERS [illegible] ROUTINE SET [illegible].

[illegible]

[illegible]

[illegible] BECAUSE SOFT PATTERNS [illegible] (HERE, A COLON) IS USED TO INDICATE THAT THE FOLLOWING CHARACTERS ARE PART OF A SOFT PATTERN A COLON ALSO ENDS A SOFT PATTERN.

A PATTERN IS DIVIDED INTO THE FOLLOWING PARTS:

<NAME, LEXICAL CLASS, OPERATOR, COUNT>

NAME: THE VARIABLE INTO WHICH THE MATCHED SUBPATTERN WILL BE STORED

LEXICAL CLASS: PARENTHESIZED STRING OF CHARACTERS

OPERATOR: EITHER < > = OR ≥ (DEFAULT >)

COUNT: INTEGER OPERAND OF THE OPERATOR (DEFAULT IS ZERO)

```
DEFINEF LOC(PARAM,LIMIT,N);    /* LOCATE PARAM */
Z = VAL(PARAM);

IF TYPE (PARAM)EQ. ≠NULL≠ THEN
     RETURN GET(LP);; /* WE ARE ALREADY HERE */
IF TYPE(PARAM) EQ. ≠SPEC≠ THEN RETURN
          /* GET LINE WITH SPECIFIED NUMBER */
     GET(DEC. Z(2:↓7));;
IF TYPE(PARAM) EQ. ≠INT≠ THEN RETURN
          /* GET LINE WITH NUMBER RELATIVE TO LP */
     GET(LP↓ DEC. Z);
     END IF TYPE;

CHECK(PARAM,≠STRING≠);  /* WE BETTER HAVE A STRING */

/* THE AMAZING TALKATIVE ARGUMENT */

IF Z(1) EQ. CHESCAPE THEN PRINT. Z(2:(↓Z)-1); READ. Z;;
HCODE = JHASH(Z);


(WHILE((GETH(HCODE,PLUS) IS. LINE) NE. LIMIT AND. PLUS) OR.
    (LINE GE. LIMIT AND. NOT. PLUS) DOING LP = LP↓1;)
    IF MATCH(GET(LINE),SETUPP(PATTERN)) THEN RETURN GET(LINE);;
END WHILE;
RETURN OM.; END LOC;


DEFINE DOBACKUP;  /* PATTERN MATCH LINEAR BACKUP */
(WHILE STRINGOK AND. J GE. 1 DOING J=J-1;)
    IF ETYP(PATTERN(J) IS. PAT) NE. 1 THEN COL = COL - ESTR(PAT);
    ELSE IF J LE. 0 THEN STRINGOK = F.;
    ELSE IF ECTR(PAT) EQ. EMAX(PAT) THEN
        COL = COL - ESTR(PAT);
```

```
        ECTR(PAT) = EMIN(PAT);
    ELSE ECTR(PAT) = ECTR(PAT) + 1;
        COL = COL + 1;
        GLOBVARS(EVAR(PAT)) = GLOBVARS(EVAR(PAT)) + STRING(COL);
    END IF;
    END WHILE;
RETURN; END DOBACKUP;


DEFINEF QUICKIE(STRING,PATTERN);
Q=EQUICK(PATTERN); LQ=↓Q-1; /* THE LONGEST HARD SUB-PATTERN */
IF LQ LE. 0 THEN RETURN T.; END IF;
RETURN ∃1<=K<=(↓STRING-LQ) ↑ ∀0<=L<=LQ ↑ STRING(K+L) EQ.Q(L+1);
END QUICKIE;



DEFINEF SUBMATCH(STRING,SUBPATT);
IF ETYP(SUBPATT) EQ. 0 THEN
    (∀1<=J<=ELEN(SUBPATT) ↑ STRING(J) EQ. ESTR(SUBPATT)(J) ) NOOP;;
    RETURN <1,J-1>;
ELSE IF EOPR(SUBPATT) EQ. ≠<≠ OR. EOPR(SUBPATT) EQ. ≠>≠ THEN
     ECNT(SUBPATT) = ECNT(SUBPATT) + 1;

 RETURN IF ∀1<=J<=ECNT(SUBPATT) ↑ STRING(J) → ELEX(SUBPATT)
    THEN <1,J> ELSE NULT.;
END IF;
END SUBMATCH;



DEFINEF SETUPP(PATTERN);
                          /* IF ANCHORING, ANCHOR ON COLUMN 1 */
IF ANCHORQ THEN VECT = <≠1↑≠+DEC.LZONE1+≠1≠>;
ELSE VECT = <(≠: :≠)>; END IF ANCHORQ;
EQK = NULC.;
(∀1<=J<=↓PATTERN) VECT = VECT + GETCOLON(PATTERN(J));;
(∀↓PATTERN>=J>=2)VECT(J-1)=<HD.VECT(J-1)+HD.VECT(J)>+TL.VECT(J-1);;
RETURN VECT+<EQK>; END SETUPP;


DEFINEF GETCOLON(PATTERN);
N = ↓PATTEN;
IF PATTERN(1) EQ. ≠1≠ AND. PATTERN(N) EQ. ≠:≠ THEN
    X = PATTERN(2:↓PATTERN-2);
    J = SPAN(X,ALPHS);
   P = <0,1,NULC.,0.NL.,≠>≠,0,0>;
    IF J NE. 0 THEN P(4) = X(1:J); X=X(J:↓X+1-J);;
    IF(↓X) EQ. 0 THEN RETURN P;;
```

```
    IF X(1) EQ. ≠(≠ THEN Z = CHARS(X(2:(BREAK(≠)≠)-1)IS.N));
        MINUSFLAG=F.;
        IF Z(1) EQ. ≠-≠ THEN MINUSFLAG = T.;Z=Z(2:↓Z-1);;
        P(5) = [+: 1<=J<=↓Z ↑ (Z(J) NE. ≠-≠ AND. Z(J-1)NE. ≠-≠)]
                                    ≤:Z(J)≥;
        Z= [+:1<=J<=↓Z ↑ (Z(J) NE. ≠-≠ AND. Z(J-1) EQ. ≠-≠)]
                                   ≤:Z(J)≥;
        IF (↓Z) GT. 1 THEN P(5) = P(5) + LEXCLASS≤ANSOB≥ - Z;;
        X = X(N+2:↓X-1-N);;
    IF (↓X) EQ. 0 THEN RETURN P;;
    IF X(1) → CHARS(≠<>=↑≠) THEN P(6) = X(1);;
    X = X(2:↓X-1);
    IF (↓X) EQ. 0 THEN RETURN P;;
    J = SPAN(X,NUMS);
    IF J NE. 0 THEN P(7) = DEC. X(1:J);P(8) = P(7);;
    IF P(6) EQ. ≠=≠ THEN P(1) = P(7);;
    RETURN P;
ELSE IF (↓PATTERN) GT. ↓EQK THEN EQK = PATTERN;;
    RETURN <↓PATTERN,0,PATTERN>;
END GETCOLON;
```

THIS ROUTINE WILL MATCH A PATTERN AGAINST SOME CHARACTER STRING AND RETURN OM. IF THE MATCH FAILS, BUT RETURN THE STARTING COLUMN AND THE FINISHING COLUMN IF IT SUCCEEDED. THE MATCHING IS STRICTLY LINEAR, NONRECURSIVE AND OF THE MOST ELEMENTARY CHARACTER. A RECURSIVE PATTERN MATCHING FACILITY WOULD PROBABLY TAKE THE EDITOR OUT OF THE REALM OF TEXT EDITORS AND INTO THE WORLD OF PROGRAMMING LANGUAGES. QBOTHER IS THE FLAG WHICH DETERMINES WHETHER IT IS WORTH ATTEMPTING A QUICKMATCH BEFORE DOING THE FULL MATCH. THIS WOULD SAVE LOCATING TIME IF THERE IS A SIGNIFICANT PERCENTAGE OF HARD SUB-PATTERNS IN THE PATTERN AS OPPOSED TO MANY SOFT PATTERNS, MOST EDIT LOCATES FALL INTO THIS CATAGORY. AT THE POINT THAT THE MAJORITY OF THE PATTERNS START BECOMING SOFT PATTERNS,THE THE USER WILL GENERALLY FIND THAT THE EDITOR IS INADEQUATE FOR HIS PURPOSES AND SHOULD START USING SNOBOL.

```
DEFINEF MATCH(STRING,PATTERN);
LENSTR = ↓STRING;
LEN = ↓PATTERN - 1;
STRINGOK=T.;
QFLAG = EQBOTHER(PATTERN);
COL=0; START = 0;
J=1;
(WHILE T.) DUMMY = DUMMY; /* CURRENT IMPLEMENTATION REQUIRES DUMMY */
    (WHILE STRINGOK AND. J LE. LEN AND. COL LE. LENSTR DOING J=J+1;)
        X=SUBMATCH(STRING(COL+1:↓STRING-COL),PATTERN(J));
        STRT=X(1); FNSH=X(2);
        IF STRT EQ. 1 THEN COL = FNSH; ELSE STRINGOK=T.; END IF;
        END WHILE;
IF COL GE. LENSTR THEN RETURN OM.;
ELSE IF J GE. LEN THEN RETURN <START,COL>;;
IF NOT. STRINGOK THEN RETURN OM.;;
    (WHILE J GT. 1 DOING J=J-1;)
    IF NOT. DOBACK(PATTERN,J) THEN RETURN OM.;;
END WHILE;
END WHILE;
RETURN <STRT,FNSH>;END MATCH;
```

/* 4.3 CHANGE PROCESS */

THIS SECTION DESCRIBES THE CHANGE PROCESS--THE MANIPULATION OF THE STRINGS WHICH TAKE PLACE AFTER THE LOCATE PROCESS HAS DETERMINED THE TWO THINGS IT WAS DESIGNED TO FIND OUT:

A. THAT THE STRING SATISFIES THE BOOLEAN PREDICATE WHICH IT SEARCHED (THIS USUALLY MEANS THAT THE STRING IT ATTEMPTED TO FIND WAS PART OF THE CURRENT LINE) AND

B. THE LIMITS OF THE SUBSTRING OF THE CURRENT LINE WHICH SATISFY THE RELATION.

THERE ARE THREE PARTS INTO WHICH THE CURRENT LINE MAY BE DIVIDED TO DISCUSS THE CHANGE PROCESS, THE INITIAL PART, THE CHANGEABLE PART,AND THE TERMINAL PART. THE CHANGE PROCESS IS RESPONSIBLE TO CONCATENATE THE PREVIOUS PART, THE CHANGE-STRING, AND THE FOLLOWING PART TOGETHER, IGNORING (DELETING) THE CHANGEABLE PART, WHICH MAY BE THE NULL STRING.THE CHANGE PROCESS MUST DETERMINE WHAT IS THE CURRENT LINE AND WHAT IS THE CHANGE-STRING(CS), THE CURRENT LINE IS FOUND IN THE GLOBAL VARIABLE CURRENT, THE CHANGE STRING IS OBTAINED BY INVOKING THE ROUTINE CSEVAL.

THE ROUTINE CSEVAL DECIDES WHAT THE CHANGE STRING IS AT A GIVEN MOMENT OF A CHANGE, SINCE THE GLOBAL VARIABLES WHICH MIGHT BE CONTAINED IN THE CHANGE STRING MAKE IT EXTREMELY DYNAMIC.CSEVAL PROCEEDS AS FOLLOWS:IF THERE ARE NO COLONS IN THE CHANGE STRING (CS) AND I AM ASSUMING THAT THE USER IS USING THE DEFAULT BREAK CHARACTER,THE COLON,THEN THE CSPRIME RETURNED BY CSEVAL IS IDENTICAL TO THE CS INPUT TO CSEVAL,IF THERE ARE ONLY DOUBLE COLONS,THEN THE RETURNED STRING IS THE SAME WITH THE EXCEPTION THAT THE DOUBLE COLONS BECOME SINGLE COLONS. IF THE CASE OF SINGLE COLONS,WE HAVE EITHER AN ERROR OR A SOFT PATTERN.CSEVAL USES THE BREAK CHARACTERS TO DIVIDE THE CHANGE STRING CS INTO SUBPATTERNS SIMILARLY TO THE LOACATE PROCESS, HOWEVER THE SUBSTITUTIONS ARE FIXED AT THE TIME OF THE CHANGE, SO THAT NO SEARCHING HAS TO BE DONE, MERELY SUBSTITUTION OF KNOWN VALUES, IF SOME GLOBAL QUANTITIY REFERRED TO AT THIS TIME IS UNKNOWN, AN ERROR INDICATION IS RETURNED TO THE USER. THE POSSIBLILITES FOR CHANGE STRING SOFT PATTERNS ARE AS FOLLOWS:

A)A SINGLE ALPHABETIC CHARACTER--THE PATTERN EVALUATES TO THE VALUE OF THE GLOBAL VARIABLE NAMED BY THE CHARACTER,

B)A SINGLE DIGIT FOLLOWED BY A SINGLE ALPHABETIC CHARACTER IN THIS CASE THE PATTERN EVALUATES TO THE VALUE OF THE

GLOBAL VARIABLE REPEATED THE NUMBER OF TIMES SPECIFIED BY THE DIGIT.

C)A SINGLE DIGIT FOLLOWED BY AN ASTERISK FOLLOWED BY A STRING OF CHARACTERS OF LENGTH LESS THAN 10 THEN THE PATTERN EVALUATES TO THAT CHARACTER STRING REPEATED THE NUMBER OF TIMES SPECIFIED BY THE DIGIT.

D)A SINGLE ALPHABETIC CHARACTER FOLLOWED BY AN ASTERISK FOLLOWED BY A STRING OF FROM 1 TO 10 CHARACTERS THEN THE PATTERN EVALUATES TO THE STRING REPEATED N TIMES WHERE N IS THE LENGTH OF THE STRING TO WHICH THE GLOBAL VARIABLE SPECIFIED BY THE LETTER EVALUATES.

E)TWO ALPHABETIC CHARACTERS IN THIS CASE THE SECOND GLOBAL VARIABLE IS REPEATED THE NUMBER OF TIMES SPECIFIED BY THE LENGTH OF THE FIRST GLOBAL VARIABLE.

EXAMPLES:

ASSUME THAT THE GLOBAL VARIABLES A AND B HAVE VALUES
A IS THE STRING ≠ABC≠ AND B IS THE STRING ≠XYZ≠

```
/:A:/   EVALUATES TO THE STRING ≠ABC≠
/:4A:/  EVALUATES TO THE STRING ≠ABCABCABCABC≠
/:4*A:/ EVALUATES TO THE STRING ≠AAAA≠
/:A*B:/ EVALUATES TO THE STRING ≠≠BBB≠
/:AB:/  EVALUATES TO THE STRING ≠XYZXYZXYZ≠
        THAT IS, ≠XYZ≠ REPEATED LENGTH(≠ABC≠) TIMES.
```

ROUTINE DOCHANGE IS THE MASTER CHANGER, IT IS CALLED BY THE MAIN INTERPRETER (ROUTINE CHANGE, WHICH DETERMINES THE ARGUMENTS AND THAT, AS CAN BE SEEN, IS MORE COMPLICATED THAN DOING THE CHANGE ITSELF, TO ALLOW FOR MANY USER-CONVENIENT SYNTACTIC DICTIONS). FIRST OF ALL THE LOCATING IS DONE, THEN CSEVAL IS CALLED TO DYNAMICALLY EVALUATE THE CHANGE STRING, THE TRIPLE CONCATENATION IS THEN CALCULATED AND REPLACED IN THE HASH TABLE:

```
NEW=NEW+STRING(START-1)+CSEVAL(SS);
```

PURE SIMPLICITY IN SETL, BUT ALLOWING FOR VARIOUS DEGREES OF EXPLICIT COMPLEX COPIES IN A LOWER LEVEL LANGUAGE, DEPENDING ON HOW WELL THE LOWER LEVEL LANGUAGE WAS DESIGNED TO HANDLE CHARACTER STRINGS. THOSE COPIES ARE NECESSARY BUT THEIR INVOCATION SHOULD REQUIRE THE MINIMUM EFFORT ON THE PART OF THE PROGRAMMER AS A MATTER OF PRINCIPLE.

```
DEFINEF DOCHANGE(OLD,N,LS,CS);
/* LS IS REPLACED WITH CS N TIMES IN OLD */
NEW = NULC.; J = 0; STRING = OLD;
(WHILE ↓STRING GT.0 AND. J LT. N DOING J=J+1;)
    DUMMY = LOCER(STRING,LS);START=DUMMY(1);LIMIT=DUMMY(2);
    IF START LE. 0 THEN RETURN OM.;;
    NEW = NEW + STRING(1:START-1)* CSEVAL(CS);
    STRING = STRING(LIMIT:↓STRING);
    END WHILE;
NEW = NEW + STRING;
IF J NE. N THEN RETURN OM.;;
RETURN NEW+STRING; END DOCHANGE;


DEFINEF CSEVAL(STRING);
STRING=(≠ ≠)+STRING;J=2;NEW=NULC.;
(WHILE J LE. ↓STRING)
     IF ≡J<=J<=↓STRING ↑ (STRING(J)EQ.≠;≠ IMP.STRING(J+1)EQ.≠;≠)
          THEN J=J;;
        (∨1<=K<=J ↑ STRING NE. ≠;≠)NEW = NEW + STRING(J);;
     IF STRING(J) EQ. ≠;≠ THEN X=STRING(J+1); J=J+2; /* ONE COLON */
        IF X→NUMS THEN      /* REPEAT STRING */
            IF STRING(J) → ALPHS AND. STRING(J+1) EQ. ≠;≠ THEN
                Z=GLOBVARS(Z); IF Z EQ. OM. THEN ERRORS(5);;J=J+2;
                NEW = NEW + (DEC. X) * Z;
            ELSE IF STRING(J) EQ. ≠*≠ AND.
                ∨J<K<=J+10↑(STRING(K)EQ.≠;≠ IMP.STRING(K+1)EQ.≠;≠)
                THEN ERRORS(6);
            ELSE IF STRING(J)EQ.≠*≠ THEN NEW=NEW+X*STRING(J+1:K);
            ELSE ERRORS(6);END IF STRING;
        ELSE Z=GLOBVARS(X);
            IF Z EQ. OM. THEN ERRORS(5); END IF Z;
            IF STRING(J+Z) EQ. ≠;≠ THEN NEW = NEW + 2;
            ELSE IF STRING(J+2) EQ. ≠*≠ AND. STRING(J+4) EQ. ≠;≠ THEN
                LENTH = ↓Z; Z=GLOBVARS(STRING(J+3));
                IF Z EQ. OM. THEN ERRORS(5); END IF Z;
                NEW = NEW + LENTH*Z;
            ELSE ERRORS(8);
            END IF STRING;
        END IF X;
    END IF STRING;
    END WHILE;
RETURN NEW; END CSEVAL;
```

```
/* 4.4 THE PAGING SCHEME FOR MEMORY ALLOTMENT */
```

THE PAGING SYSTEM OF EDIT IS SIMPLE AND DIRECT. EACH PAGE CONSISTS OF LPAGE (SAY 64) CARDS. ONE PAGE OF TEXT IS IN MEMORY AT A TIME, AN ADDITIONAL PAGE IN MEMORY STARTS EMPTY. CORRECTIONS ARE ADDED TO THIS PAGE UNTIL LPAGE CORRECTIONS HAVE BEEN ACCUMULATED. AT THAT POINT, THE ENTIRE PAGE IS WRITTEN ONTO DISK. TO COMPLICATE THE PROGRAM, IT MAY BE DESIRED TO MAKE THE PAGES DIFFERENT SIZES, BUT THAT IS NOT CONSIDERED IN THIS SETL PROGRAM. THE NYU EDITOR ACTUALLY HAS TWO DIFFERENT PAGE SIZES AND ITS AUTHORS CLAIM THIS SAVES FIELD LENGTH.

<TABLEENTRY> = <PAGENUMBER,LINENUMBER,HASHCODE>;

EXAMPLE: HASHCODE 42 BITS
PAGENO 12 BITS = 4096
LINENO 6 BITS = 64
TOTAL 60 BITS
4096X64= 262144 CARDS MAX BEFORE EDIT OVERFLOW

BECAUSE THE SETL VERSION IS SET UP COMPLETELY INDEPENDENTLY OF WORD-SIZE, EXPERIMENTATION WITH DIFFERENT LENGTH HASHCODES AS WELL AS WITH THE TRADE-OFF BETWEEN MAXPAGES AND LPAGE WILL BE VERY EASY. OF COURSE, IT IS OBVIOUS THAT THE LAST TWO ARE INVERSELY PROPORTIONAL AND TIME DECREASES WITH LPAGE WHILE MEMORY USED INCREASES. IN MOST OPERATING WHILE MEMORY USED INCREASES. IN MOST OPERATING ENVIRONMENT MOST EDITING IS DONE, SPACE IS USUALLY MORE CRITICAL THAN TIME.

SUBROUTINE PUT WRITES THE CURRENT LINE INTO THE CORRECTION PAGE PLACING ITS POINTER AND HASHCODE INTO THE HASH TABLE AS ENTRY NUMBER LP. IF THE CORRECTIONS PAGE OVERFLOWS, IT IS WRITTEN ONTO THE DISK BEFORE WRITING THE CURRENT LINE. ALL LINES IN IN THE HASH TABLE AFTER LP ARE PUSHED DOWN ONE ENTRY TO MAKE ROOM FOR THE NEW LINE, THUS CHANGING THE NUMBERING OF ALL THE LINES AFTER LP. IN SOME INSTANCES THIS AN INCONVENIENCE BUT THE NYU EDITOR WAS BUILT AROUND THIS ASSUMPTION PROBABLY BECAUSE OF EFFICIENCY. IN A SETL ENVIRONMENT PERHAPS EFFICIENCY CAN BE PARTIALLY IGNORED, BUT THE FEATURE OF CONSTANT NUMBERS FOR LINES WAS NOT IGNORED, BUT RATHER PURPOSELY LEFT OUT OF THIS PROGRAM.

```
DEFINE PUT(CARD);
IF PUTUPTO EQ. LPAGE THEN WRITER(#ZZZZZVE#,LASTPAGE,PAGEB);
    PUTUPTO = 0; LASTPAGE = LASTPAGE + 1;
    /* IF EDITFILE OVERFLOW THEN GARBAGE COLLECT THE DISK SPACE
       USED BY SAVING AND EDITING THE SAVED FILE. */
    IF LASTPAGE GT. MAXPAGES THEN ERROR(9);
    SAVEIT(FILENAME);
        IF LASTPAGE GT.(MAXPAGES-1) THEN ENDEDIT;
                    ELSE EDIT(FILENAME));;
        END IF LASTPAGE;
    PUTUPTO = PUTUPTO + 1;
    END IF PUTUPTO;
TABLE(LP) = <LASTPAGE,PUTUPTO,IHASH(CARD)>;
IF UPDING AND. NOT. INITFLAG THEN CARD = CARD(1:80) MIN. ↓ CARD;;
PAGEB(PUTUPTO) = TRUNC. CARD; RETURN; END PUT;
```

THE ROUTINE GET(I)IS USED WHEN YOU WANT TO USE THE CARD NUMBER I IN THE FILE. GET WILL DECIDE IF THE LINE IS IN ONE OF THE PAGES IN CORE AND IF NOT, IT WILL READ IN THE CORRECT PAGE AND UNPACK THE STRING VALUE AND ADD IN THE TRUNCATED BLANKS UP TO IZONE. LOCATE L3 COULD THEN BE CODED AS PRINT. GET(3);

```
DEFINEF GET(I);
I = LINES MIN. I MAX. 1; /* NOTHING IS OUT OF RANGE */
IF I EQ. CURRENTN THEN RETURN CURRENT;;
PAGE=PAGENO (TABLE(I)); LINEP = LINENO(TABLE(I));
IF GETPAGE(PAGE) EQ. 0 THEN RETURN OM.; END IF;
/* THE CORRECT PAGE IS NOW IN MEMORY */
```

```
CURRENT=UNPACK(PAGE,LINEP); CURRENT = CURRENT + (150-↓CURRENT)*(≠ ≠);
LP = I; CURRENTN = I;
RETURN CURRENT; END GET;
```

```
                    THE  FUNCTION GETH(HASH, PLUS) FINDS THE NEXT LINE
                    WHICH HAS  EVERY BIT IN ITS HASH CODE SET WHICH IS
                    SET IN HASH AND RETURNS ITS LINE NUMBER.IF PLUS IS
                    F, THEN THE SEARCH IS DONE BACKWARDS. SEARCH FAILS
                    IF TOP OR BOTTOM OF TABLE IS REACHED.

DEFINEF GETH(HASH,PLUS); I=LP;
(WHILE I LE. LINES AND. I GT.0 AND.
   NOT. HASHCD(TABLE(I)) INCS. HASH)
    IF PLUS THEN I=I+1;ELSE I=I-1)END IF;
END WHILE;
IF I GT. LINES OR. I LE. 0 THEN I = 0; ELSE LP=I; END IF I;
RETURN I; END GETH;
```

```
                    GET THE NEXT PAGE OF TEXT OFF
                    THE RANDOM PAGE FILE.

DEFINEF GETPAGE(PAGE);
IF PAGE GT. LASTPAGE THEN RETURN F.;;  /*PAGE TOO BIG*/
IF ≡1<=K<=↓PAGES ↑ PAGES(K) EQ. PAGES THEN RETURN K; END IF;
PAGEBODY(PAGE)=READR(PAGEFILE,PAGE);
IF PAGEBODY(PAGE) NE. OM. THEN
    TL. PAGES = TL. TL. PAGES + <PAGE>; RETURN ↓PAGES;
ELSE RETURN 0; END IF;
END GETPAGE;
```

```
DEFINEF UNPACK(PAGE,LINEP);
STRING=(PAGEBODY(PAGE))(LINEP)+150*(≠ ≠);
RETURN STRING(1:LZONE2);
END UNPACK;
```

AN APPROXIMATE MAP OF MEMORY ALLOCATION FOR EDIT:

```
**************************************************************
*     PROGRAM AND I/O BUFFERS
**************************************************************
*     CURRENT PARAMETERS
**************************************************************
*     CURRENT PAGES OF TEXT
**************************************************************
*     CURRENT CORRECTIONS PAGE
**************************************************************
*     POINTERS AND STRINGS (A FIXED SIZE AREA)
**************************************************************
*     MACRO TABLE
**************************************************************
*     CURRENT PAGE OF THE HASH TABLE--ONE ENTRY PER LINE
**************************************************************
```

IHASH AND JHASH WHICH CALCULATE THE ACTUAL HASH CODE OF A HARD AND MIXED STRING (SOFT STRINGS AUTOMATICALLY HAVE A HASH CODE OF ZERO) ARE FOUND IN THE SECTION CALLED UTILITY ROUTINES.

DEL DELETES TABLE ENTRIES A THROUGH B, THUS DELETING THESE LINES FROM ALL FUTURE SAVE FILES. TABLE MANAGEMENT, OF COURSE, OCCURS WITHOUT THE USER (OR IN THIS CASE EVEN THE PROGRAMMER) EVER KNOWING ABOUT IT.

```
DEFINE DEL(A,B);
TABLE = TABLE(1:A-1)+TABLE(B+1:*TABLE);RETURN;END DEL;
```

```
        /* 5.  AN INTERACTIVE MINI-OPERATING SYSTEM */

                /* 5.1 O.S. ROUTINES */


DEFINE OSINIT;  /* A LIST OF OPERATING SYSTEM COMMANDS */
COMMANDS = S;
<≠BACKSPACE≠,ZBACKSP>,
<≠BATCH≠     ,BATCH>,
<≠CLOSE≠     ,CLOSE>,
<≠COPY≠      ,ZCOPY>,
<≠EDIT≠      ,EDITOS>,
<≠FILES≠     ,FILES>,
<≠LOGIN≠     ,LOGIN>,
<≠LOGOUT≠    ,LOGOUT>,
<≠OPEN≠      ,ZOPEN>,
<≠REWIND≠    ,REWINDF>,
<≠SKIP≠      ,ZSKIP> ≥;
RETURN; END OSINIT;


DEFINE ZBACKSP;BACKSPAC(FNT(VAL(ARG1)));RETURN;END ZBACKSP;


DEFINE BATCH;
                BATCH(FILE,DISPOSITION) WHERE FILE IS AN OPEN FILE.
                VALID DISPOSITIONS ARE:

                   BATCH(FILE,PRINT)
                   BATCH(FILE,PUNCH)
                   BATCH(FILE,RENAME,NEWNAME)

X = FNT(VAL(ARG1)); IF X EQ. 0D. THEN ABORTFLAG=T.; RETURN;;
CHECK(ARG2,≠NAME≠); Z = VAL(ARG2);
IF Z EQ. ≠PRINT≠ THEN COPYF(X,≠OUTPUT≠); EVICT(X);,RETURN;;
IF Z EQ. ≠PUNCH≠ THEN COPYF(X,≠PUNCH≠); EVICT(X); RETURN;;
IF Z EQ. ≠RENAME≠ AND. VALID. (VAL(ARG3) IS,X) THEN
    FNT(VAL(ARG)) = X; FNT LESF. VAL(ARG1); RETURN;;
PRINT. ≠ILLEGAL DISPOSITION≠; ABORTFLAG=T.; RETURN; END BATCH;


DEFINE CLOSE; EVICT(VAL(ARG1)); RETURN; END CLOSE;


DEFINE ZCOPY; IF VALID. ARG1 AND. VALID.ARG2 THEN
    COPYF(VAL(ARG1),VAL(ARG2));ELSE ABORTFLAG = T.;;
```

```
RETURN;END ZCOPY;


DEFINE EDITOS;
INTROEDIT; EDITING = EDIT(VAL(ARG1),VAL(ARG2));
ABORTFLAG=EDITING;RETURN; END EDITOS;


DEFINE FILES;
PRINT, ≠LOCAL FILES≠; (∨X→FNT)PRINT, HD, X;; RETURN; END FILES;


DEFINE KEEP;
Z=NL,;(∨1<=K<=↓ARG) Z WITH, VAL(ARG(L)); END ∨;
(∨X→FNT ↑ NOT, HD, X → Z) RETURNF(X);END ∨;
RETURN; END KEEP;


DEFINE LOGIN;PRINT,≠PREVIOUS USER STILL LOGGED IN≠;ABORTFLAG=T.;
RETURN; END LOGIN;


DEFINE LOGOUT; FILES; FNT = NL.;
IF EDITING THEN ENDEDIT;; LOGGEDIN = F.;
PRINT, ≠YOU NO LONGER OWN ANY PRIVATE FILES≠;
PRINT, ≠CP TIME ≠,DEC,RANDOM(1500)-500+≠.≠ + DEC,RANDOM(1000);
PRINT, ≠PP TIME ≠,DEC,RANDOM(1000)+≠.≠+DEC,RANDOM(1000);
PRINT, ≠LOGGED OUT AT≠, TIME; RETURN; END LOGOUT;


DEFINE ZOPEN;OPENF(VAL(ARG1));RETURN; END ZOPEN;


DEFINE ZREWIND;
IF NOT, VALID, ARG1 THEN ILLARG;;
REWINDF(VAL(ARG1)); RETURN; END ZREWIND;


DEFINE ZSKIP;
SKIPF(VAL(ARG1),VAL(ARG2));RETURN; END ZSKIP;


DEFINE XEQ(FILE);
                    XEQ  SIMULATES EXECUTION  OF A BINARY  LOAD AND GO
                    FILE
IF VALID, FILE THEN PRINT, FILE,≠EXECUTED≠;
ELSE PRINT, ≠FATAL LOAD ERRORS≠;; RETURN ; END XEQ;
```

```
/* 5.2 MAIN LOOP  */

        THE  EDIT PROMPT  IS ≠E>≠. THIS INDICATES THAT THE
        USER IS  IN EDIT MODE. WHEN THE USER IS IN COMMAND
        MODE, THE OPERATING  SYSTEM  GIVES  THE  PROMPT
        ≠COMMAND>≠. WHEN THE USER IS NOT YET LOGGED IN,THE
        SYSTEM GIVES THE PROMPT ≠TYPE LOGIN≠.


DEFINE MAINLOOP; (WHILE POWERON)    /* START AN INFINITE LOOP */

(WHILE NOT.LOGGEDIN)  /* THE FOLLOWING CODE LOGS IN A USER */
    NAME = NULC.;
    (WHILE NAME(1:6) NE. ≠LOGIN.≠)PRINT. ≠PLEASE LOGIN≠; READ. LINE;;
    PRINT. ≠WHAT IS YOUR NAME>≠; READ. NAME;
    PRINT.≠PASSWORD>≠; READ. LINE;
    IF LINE(1:4) NE. ≠SETL≠ THEN PRINT. ≠INVALID PASSWORD≠;
    ELSE LINE = NULC.; LOGGEDIN = T.;
         PRINT. NAME, ≠LOGGED IN AT≠, TIME ; END IF LINE;NAME=NULC.;
    END WHILE NOT. LOGGEDIN;

/* THE USER IS NOW LOGGED IN*/

IF REEDIT THEN EDITING = T.; REEDIT = F.; END IF REEDIT;
ABORTFLAG = F.;

IF EDITING THEN ABORTFLAG=EDITLOOP;; /* FIRST TRY AN EDIT COMMAND */

IF NOT. EDITING OR. ABORTFLAG THEN KEYWORD = RECEIVE;
    IF KEYWORD EQ. CM. THEN PRINT.≠CONTROL CARD ERROR≠;ABORTFLAG=T.;
    ELSE IF (COMMANDS(KEYWORD)IS.X) EQ. CM. THEN XEQ(KEYWORD);
    END IF;
    IF ABORTFLAG THEN PRINT. ≠ABORTED≠;;
    ABORTFLAG=F.;
    END IF NOT. EDITING;

END WHILE POWERON;

RETURN; END MAINLOOP;
```

```
DEFINEF EDITLOOP;  /* ATTEMPT TO EXECUTE AN EDIT COMMAND */
IF NOT. EDITING THEN RETURN F.; END IF NOT. EDITING;

IF SKIPPING THEN CARD = NULC.;
    (WHILE SKIPPER → LABELS(CARD) OR. SKIPSTOP → LABELS(CARD))
        CARD = RECEIVE;END WHILE;
    SKIPPING = F.;END IF SKIPPING;

/* READ A KEYWORD FROM THE INPUT BUFFER */
CARD = RECEIVE; SCAN. CARD;

/* SEE IF THERE IS ENOUGH TIME LEFT */
IF NOT. TIMELEFT THEN RETURN F.; END IF NOT.;

/* AN INTEGER MEANS LINE REPETITION */
IF TYPE(X) EQ. ≠INT≠ THEN
    Z = (DEC.X)-1; LP = LINES MIN. LP+1;
    IF Z LE. 0 OR. LP LE. 1 OR. LP GE. LINES THEN RETURN F.;END IF Z;
/* OR. LINE IS EMPTY */
/* WHAT DOES RESET DO AGAIN ≥≥≥≥≥≥≥≥≥≥≥≥≥≥≥≥≥≥≥ */
    REPLACE. DEC. Z; RESET; RETURN T.; END IF TYPE;

/* STAR MEANS INFINITE REPETITION */
IF TYPE(X) EQ.≠STRING≠ AND. X EQ. ≠*≠ THEN
    INFIT = T.;     /* INFINITE ITERATION */
    IF LP EQ. 0 OR. LP EQ. LINES OR. NOT INFIT THEN
        INFIT = F.;ERROR(10); END IF LP;
    LP = LINES MIN. LP+1; RESET; RETURN T.; END IF TYPE;
/* NOW REPEAT THE CURRENT LINE */

IF (↓X) LE. 0 THEN PRINT. ≠BAD CARD≠;RETURN T.; END IF;

/* RECOGNIZE AND HANDLE MACRO DEFINITIONS */
IF MACROS(X) NE.ON. THEN ARG = MACLEARGS;
DOMACRO(ARG,X); RETURN T.; END IF MACROS;  /* TO THE MAIN LOOP */

/* INFORM USER OF READ ACTIVITY */
IF ALTRDING AND. NOTSENT THEN PRINT. ≠READING≠,READNAME;
    NOTSENT = F.; END IF ALTRDING;

/* WE TRY FOR AN EDIT COMMAND */
IF ABBREV(X) IS.Y NE.ON. AND.
    (CHAR EQ. (≠ ≠) OR.CHAR EQ. ≠;≠)
THEN OKFLAG=T.; Z=EXECUTES(Y);Z; RETURN OKFLAG;
ELSE RETURN T.; REEDIT = T.; EDITING = F.; END IF ABBREV;
END EDITLOOP;
```

/* 5.3 LEXICAL PROCESSING */

SYNTAX OF OPERATING SYSTEM COMMANDS

```
<SYSTEM-COMMAND>  ->   <*NAME>
                  ->   <SYSTEM-COMMAND><*SEP><ARG>
<ARG>             ->   <ARGUMENT>
                  ->   <*NAME>  =  <ARGUMENT>
<ARGUMENT>        ->   <*INT>
                  ->   <*NAME>
                  ->   <*NULL>
```

LEXICAL TYPES ARE:

```
*CHAR          ANY SINGLE CHARACTER
*CHARS         ANY CHARACTERS NOT QUOTES OR SEPS
*INT           6 OR FEWER DIGITS
*NAME          7 OR FEWER ALPHANUMERICS STARTING WITH ALPH
*NULL          NO CHARACTER
*PLUSMI        PLUS OR MINUS SIGN
*QUOTE         ANY(SLASH, QUOTE,QUESTION MARK, ETC.)
*SEP           COMMA OR BLANK
```

SYNTAX OF EDIT COMMANDS

```
<EDIT-COMMAND> -> <KEYWORD>
               -> <EDIT-COMMAND><ARG>
<ARG>          -> <*SEP> <ARGUMENT>
               -> <ARGUMENT>
<ARGUMENT>     -> <LOCARG> <*PLUSMI> <LOCARG>
               -> <LOCARG>
<LOCARG  >     -> <*QUOTE> <*NOL-QUOTES> <*QUOTE>
               -> L <*INT>
               -> <*INT>
               -> * <*NAME>
               -> <*NAME>
               -> <*NULL>
```

AN EDIT KEYWORD IS A GROUP OF ALPHABETIC CHARACTERS ON THE LIST OF EDIT-RECOGNIZED KEYWORDS. EDIT ARGUMENTS HAVE ONE OF THESE TYPES--INTEGER,NAME,STRING,LINE SPECIFYER,OR NULL. THE BLANK THAT SEPARATES TWO ARGUMENTS MAY BE REPLACED BY A COMMA. TWO COMMAS IN A ROW INDICATES A NULL ARGUMENT,AS DOES THE LACK OF ANY FINAL ARGUMENTS. SINCE ARGUMENTS OF TYPE STRING ARE DELIMITED BY QUOTES, THEY DO NOT REQUIRE BLANKS BETWEEN THEM.

EDIT COMMANDS ARE TERMINATED BY A SEMICOLON OR AN END OF

```
        LINE.AN UNQUOTED PERIOD IMMEDIATELY DECLARES IT TO BE SYSTEM
        COMMAND,NOT AN EDIT COMMAND,AND THE STATEMENT IS TURNED OVER
        TO THE OPERATING SYSTEM FOR POSSIBLE EXECUTION.

        VARIOUS CHARACTERS MAY BE USED AS QUOTE MARKS INCLUDING THE
        QUOTE(4-8PUNCH),THE SLASH,THE QUESTION MARK(12-5-8).THE SAME
        CHARACTER MUST CLOSE THE QUOTE.


NAMES CONSIST OF AN ALPHABETIC FOLLOWED BY FROM 0 TO 6 ALPHANUMERICS.

 STRINGS CONSIST OF A QUOTE MARK, AN ARBITRARY SIZE CHARACTER STRING
FOLLOWED BY THE MATCHING QUOTE MARK.

 THE TYPE CHAR MERELY MEANS A STRING OF LENGTH ONE AND IS TESTED FOR
ACCORDINGLY.

$  THE TYPE NUMB IS A SPAN OF 1 TO 4 DIGITS.

  THE TYPE PLUSMI IS A  PLUS OR MINUS SIGN USED  AS A UNARY OPERATOR
APPLIED TO THE IMMEDIATELY FOLLOWING OPERAND.

  THE TYPE NULL MEANS EITHER THAT THERE ARE NO FURTHER ARGUMENTS, OR
THAT TWO CONSECUTIVE UNQUOTED COMMAS APPEARED.

ARGTYPES =≤;<0,≠INT≠>,
            <1,≠NULL≠>,
            <2,≠STRING≠>,
            <3,≠NAME≠>,
            <4,≠PLUSMI≠>,
            <5,≠CHAR≠> ≥;


DEFINEF LABELS(STRING);
LABEL = NL.;
(WHILE(BREAK(STRING,≠:≠)IS.J) NE. 0 AND.
    STRING((SPAN(STRING,≤;≠ ≠≥)+1) IS. ; ) → ALPHANUM )
    STRING(K:J+1-K) IN. LABEL;
    STRING = STRING(J+1:↓STRING+1-J);
    END WHILE;
RETURN LABEL; END LABELS;


DEFINEF NXTSCAN(LINE);  /* BREAK A COMMAND OFF A LINE */

LEN = ↓LINE; J=J-LEN;
/* FIND A PERIOD, A QUOTE, OR A SEMICOLON, WHICHEVER COMES FIRST */
K=0;
(WHILE K LE. LEN)
K=BREAK(LINE,LINE(K+1:LEN-K),QUOTESET+≤; ≠;≠ . ≠,≠ ≥);
```

```
IF LINE(K) EQ. ≠;≠ OR. LINE(K) EQ. ≠,≠ OR. K GE. LEN
THEN RETURN LINE(1:K);
END IF LINE;
L=FINDCHAR(LINE(K+1:LEN-K),LINE(K));
IF K EQ. 0 THEN RETURN LEN + J;
ELSE IF L EQ. 0 OR. L EQ. ↓LEN THEN RETURN LEN+J; END IF;
K=K+L;
END WHILE;
RETURN LEN+J; END NXTSCAN;


DEFINEF RECEIVE;  /* GET THE NEXT COMMAND FROM WHEREVER*/
(WHILE RECLINE EQ. OM.) /* PREVIOUS LINE BLANK */
    IF (↓MACSTAK IS.K) GT. 0 THEN /* STACKED COMMAND */
        RECLINE = MACSTAK(K); MACSTAK(K) = OM.;
    ELSE RECLINE = ER; FLAG = T.; /* READ FROM TOP INPUT FILE */
        (WHILE RECLINE EQ. ER AND. INPUTSTAK NE. OM.)
            IF FLAG THEN FLAG=F.;
            ELSE INPUTFILE = GET(INPUTSTAK);
            END IF FLAG;
            END WHILE;
        IF INPUTSTAK EQ. OM. THEN ENDEDIT; STOP;; /* BATCH ONLY */
    END IF;

                NOTE THAT BY REMOVING NULL STATEMENTS LEXICALLY,WE
                CAN SAVE EXECUTION TIME.THIS IS ESPECIALLY TRUE OF
                THE COMMON OCCURANCE OF A REDUNDANT SEMICOLON
                FOLLOWING THAT LAST STATEMENT OF A LINE.

CARD = NXTSCAN(RECLINE);
RECLINE = RECLINE(↓CARD+1:↓RECLINE);
END WHILE RECLINE;
IF (≠I≠) → LISTSET OR. (≠H≠) → LISTSET ALD. HACKING OR.
    (≠R≠) → LISTSET AND. READING
THEN PRINT. ≠COMMAND> ≠,CARD; END IF;
RETURN CARD; END RECEIVE;


DEFINEF SCAN. STRING; /* RETURN KEYWORD AND CRACK ARGUMENTS */
ARG = NULT.; KEYWORD = NULC.;
RETURN KEYWORD; END SCAN.;


DEFINE TRANSMIT.LINE; MACSTAK=LINE+MACSTAK; RETURN; END TRANSMIT.;
```

```
        /* INTEGER ARGUMENTS TO EDIT COMMANDS */

DEFINEF DELARG(J); /* THE TWO ARGUMENTS TO THE DELETE COMMAND */
/* DELETE,ZEXECUTE,PRINT,REPLACE ENTER WITH J = 0
   ALIGN,OVERLAY,SAVE,SPLIT WITH J = 1 */
IF NOT, OKFLAG THEN RETURN 0; END IF;
IF J NE.0 AND.J NE.1 THEN ERRORS(3);OKFLAG=F.; END IF J;
X = ARG(1+J); Z = ARG(2+J);
IF TYPE(X) EQ. ≠NULL≠ THEN /* DELETE ONLY CURRENT LINE */
    M1 = LP IS. M2; RETURN T.; END IF;
IF TYPE(Z) EQ. ≠NULL≠ THEN /* DELETE DOWN TO THE FIRST ARGUMENT */
    Z = X; M1 = LP;
ELSE M1 = LOC(X,0); END IF; /* DELETE FROM FIRST ARG */
IF M1 EQ. 0 THEN RETURN F.; END IF;
PUSH(LP); LP = M1;
M2 = LOC(Z,0); POP(LP); /* DELETE DOWN TO SECOND ARGUMENT */
IF M2 EQ. 0 OR. M2 LT. M1 THEN ERROR(2);OKFLAG=F.; END IF;
RETURN SAFE(M1,M2); END DELARG;


DEFINEF ZONEARG;
/* USED FOR ARGS TO ZONE, PZONE, MZONE AND LZONE(CZONE)
   ZONE IS SYNONYMOUS WITH LZONE AND PZONE */
M1 = IF TYPE(ARG1) EQ. ≠INT≠ THEN VAL(ARG1) ELSE 1;
M2 = IF TYPE(ARG2) EQ. ≠INT≠ THEN VAL(ARG2) ELSE 72;
IF M1 GT. M2 THEN ERROR(2);OKFLAG=F.; END IF M1;
IF M1 LE. 0 OR. M2 GT. 150 THEN ILLARG; END IF M1;
RETURN T.; END ZONEARG;
```

/* 5.4 MACROS */

THE MACRO PREPROCESSOR IS SIMPLIFIED BY A NUMBER OF BASIC ASSUMPTIONS AS FOLLOWS.

A)MACRO DEFINITIONS MAY NOT APPEAR WITHIN MACRO DEFINITIONS.

B)EDIT COMMANDS MAY NOT OVERLAP CARD BOUNDARIES.(THUS THE NUMBER OF MACRO ARGUMENTS IS LIMITED BY THE SIZE OF THE DEVICE INPUT LINE.

C)BOTH MACRO DEFINITION AND MACRO SUBSTITUTION ARE TRIGGERED NOT ONLY AT TOKEN BOUNDARIES, BUT ALSO ONLY AT THE BOUNDARY OF A STATEMENT. (THIS MEANS THAT A MACRO CAN ONLY BE A LIST OF EXECUTABLE COMMANDS.

D)RECURSIVE MACRO EXPANSION IS NOT DONE RECURSIVELY BUT BY REPEATED LINEAR SCANNING USING THE DISK AS THE STACK, ALLOWING EXTREMELY SIMPLE SOFTWARE ROUTINES AND TREMENDOUSLY LARGE MACROS WITH VERY LITTLE CORE MEMORY REQUIRED THOUGH REDUCING SPEED BY A LARGE BUT UNNOTICEABLE FACTOR. IT IS UNNOTICED BECAUSE THE REAL TIME RESPONSE OF A TIMESHARING ENVIRONMENT IS MEASURED IN SECONDS WHILE THE WORST CASE RESPONSE OF THE DISK ON A 6600 SYSTEM WITH MANY TIMESHARING USERS IS MEASURED IN SECONDS. THAT IS, EVEN IN SUCH A WORST CASE EXAMPLE, MACRO EXPANSION FROM THE DISK BASE WILL ALWAYS BE FASTER THAN TYPING IN THE COMMANDS BY HAND.

E) THE EXISTENCE OF RANDOM-ACCESS DISK FILES.

```
DEFINE DOMACRO(ARG,X);

/* YOU MAY ARBITRARILY HAVE ONLY 10 MACRO ARGS */

(X↓ARG<I<=10)ARG(I)=NULC.; END X;
LINE = ≠ ≠←READR(MACROS.X); NEW = NULC.; COL = 2;
(WHILE ΞCOL<=COL<=↓LINE↑ LINE(COL) EQ. CHESCAPE)
    NEW = NEW + LINE(1:COL-1) +
        IF LINE(COL)EQ.CHESCAPE THEN CHESCAPE
        ELSE ARG(N);
    END WHILE; COL = COL + 2;
TRANSMIT, LINE;
RETURN; END DOMACRO;
```

THIS ROUTINE WILL SUBSTITUTE THE ARGUMENT FLAG FOR EACH OCCURANCE OF A MACRO ARG IN THE MACRO TEXT AND WRITE THE TRANSFORMED TEXT ONTO A RANDOM FILE.

```
DEFINE MAKMACRO(MNAME,MARGS);
CARD = RECEIVE;
(WHILE CARD(1) NE. CHESCAPE DOING CARD = RECEIVE;)
    CARD = (≠ ≠)+CARD+(≠ ≠);M=2;
    (WHILE ∃M<=M<=↓CARD↑(NOT.CARD(M-1)→ALPHANUM AND.
           ∃1<=K<=↓ARG↑(NOT.CARD(M+1+↓MARGS(K) IS. Z)→ALPHANUM AND.
                        CARD(M:Z) EQ. MARGS(K) )))
        CARD = CARD(1:IF CARD(M-1)EQ.CHCONCAT THEN M-2 ELSE M-1) +
               (≠ ≠) + CHESCAPE + ALPH(K) + CARD(M+Z:↓CARD);
        END WHILE ∃ M;
    WRITER(MACROS,MNAME,CARD(2:(↓CARD) -1));
    END WHILE CARD;
WRITER(MACROS,MNAME,≠MS≠);  /* END UP WITH A MACRO STOP COMMAND */
WRITER(MACROS,MNAME,ER);    /* WRITE END-OF-FILE */
RETURN; END MAKMACRO;
```

```
/* 6.1 FIRST LEVEL COMMANDS */


DEFINE ALN; IFEMPTY; J = VAL(ARG1);
IF TYPE(ARG1) EQ. ≠INT≠ AND. DELARG(1) THEN
    (v M1<=LP<=M2) PUT(ALN(GET(LP),J));END v;
                   ELSE OKFLAG = F.; END IF;
RETURN; END ALN;
```

THE ≠ANCHOR≠ COMMAND IS SYNONYMOUS WITH THE ≠LOCATE≠ COMMAND EXCEPT THAT THE ANCHORO FLAG IS SET TO SIMULATE ANCHORING AS WITH A SOFT PATTERN SUCH AS ≠:↑1:≠

```
DEFINE ANCHOR; ANCHORO = T.; LOCATE; RETURN; END ANCHOR;
```

THE SUBROUTINE ≠ASK≠ MERELY CALLS ON THE OPERATOR ≠ASK≠ TO ASK THE QUESTION WHICH IS THE ARGUMENT.

```
DEFINE ASK;
IF NOT,ASK.(IF TYPE(ARG1) EQ.≠STRING≠ THEN VAL(ARG1) ELSE ≠EXECUTE≥≠)
THEN DOSKIP(ARG2);; RETURN; END ASK;
```

CHARACTER,FOR WHICH YOU SHOULD SEE THE EXPLANATION OF THE ≠OVERLAY≠COMMAND BELOW.

```
DEFINE ZBLANK;
IF TYPE(ARG1) EQ.≠STRING≠THEN CHBLANK=(VAL(ARG1))(1:1);;
RETURN;END ZBLANK;
```

BOTTOM PLACES THE LINE POINTER AT THE BOTTOM OF THE FILE EITHER FOR EXTENSION PURPOSES OR PERHAPS TO SET UP FOR A BACKWARDS SEARCH. DO NOT CONFUSE THIS COMMAND WITH THE LINE SPECIFYER ≠B≠ USED IN LOCATE GROUPS, EVEN THOUGH THE COMMAND ≠BOTTOM≠CAN BE ABBREVIATED ≠B≠. THEY ARE CONTEXTUALLY UNAMBIGUOUS.

```
DEFINE BOTTOM; IFEMPTY; LP= LINES; RETURN;END BOTTOM;
```

IF YOU ARE CCING THEN THE FIRST COLUMN OF EACH

LINE IN YOUR FILE IS INTERPRETED AS A CHARACTER CONTROL, IE 1 MEANS A PAGE EJECT AND 0 MEANS A LINE SKIP. THIS IS USEFUL FOR A PAGED FILE, BUT ON A TELETYPE UNIT, IT IS IGNORED BY MOST OPERATING SYSTEMS.

```
DEFINE CC; CCING = T.; RETURN ;END CC;
```

THE MASTER ≠CHANGE≠ ROUTINE DECODES THE VARIOUS ARGUMENTS AND CALLS ON ≠LOC≠AND ≠DOCHANGE≠ TO FIND AND CHANGE THE REQUIRED STRINGS, FOLLOWING THE DEFINITION IN THE INTRODUCTION.

```
DEFINE CHANGE; IFEMPTY; OKFLAG = F.;
P1 = IF TYPE(ARG1) EQ. ≠NULL≠ THEN LS
ELSE IF TYPE(ARG1) EQ. ≠STRING≠ THEN VAL(ARG1) ELSE OM.;
P2 = IF TYPE(ARG2) EQ. ≠NULL≠ THEN CS
ELSE IF TYPE(ARG2) EQ. ≠STRING≠ THEN VAL(ARG2) ELSE OM.;
P3 = IF TYPE(ARG3) EQ.≠NULL≠THEN 1
ELSE IF VAL(ARG3)EQ.≠*≠ THEN LINELEN
ELSE IF TYPE(ARG3) EQ. ≠INT≠ THEN VAL(ARG3) ELSE OM.;
IF P1 EQ. OM. OR. P2 EQ. OM. OR. P3 EQ. OM. THEN ILLARG;
RETURN;END IF;
 /* DETERMINE THE FOURTH PARAMETER */
P4= LOC(ARG4,LINES); IF P4 EQ. 0 THEN RETURN; END IF;
P5 = LOC(P1,P4);
                              IF P5 NE.0 THEN KFLAG=T.;DOFCHG;END IF;
RETURN; END CHANGE;
```

THE COMMAND ≠COMMENT≠ IS USUALLY ONLY USED WHEN READING AN EXTERNAL FILE. IT MERELY PRINTS OUT ITS ARGUMENT. THUS, IN AN EXTERNAL FILE CONTAINING 6000 EDIT COMMANDS, YOU MIGHT WISH TO MAKE THE 3001 COMMAND:

```
 COMMENT,≠YOU HAVE EXECUTED 3000 LINES≠
```

THIS PRINTS OUT THE MESSAGE AT THE TELETYPE, SO THAT THE USER WOULD BE ABLE TO FEEL ASSURED THAT THE EDITOR IS ACTUALLY EDITING, YOU COULD ALSO HAVE A MACRO PRINT OUT A COMMENT IMMEDIATELY BEFORE INFORMATION IS TO BE ENTERED, EG:

```
COMMENT,≠ENTER A 2-DIGIT NUMBER≠
INSERT,≠MY AGE IS $≠
CHANGE,≠$≠,~WHAT IS YOUR AGE~
```

```
DEFINE COMMENT;PRINT.VAL(ARG1);RETURN;END COMMENT;
```

≠DELETE≠ FIRST DECODES ITS ARGUMENTS IF ANY AND FORMATS THEM INTO A STANDARD REQUEST TO THE LOWER LEVEL ROUTINE DEL WHICH DOES THE ACTUAL DELETING.

```
DEFINE DELETE; IFEMPTY;
IF DELARG(0) THEN DEL(M1,M2); ELSE CKFLAG=F.;;
RETURN; END DELETE;

DEFINE DEVICE;IF TYPE(ARG1)EQ.≠INT≠ THEN LDEVICE = VAL(ARG1);;
RETURN; END DEVICE;

DEFINE DISPLAY;
/* WE MUST CREATE THIS SET FROM SCRATCH EACH TIME DISPLAY IS
   CALLED.  THIS IS ANOTHER ARGUMENT FOR ALLOWING POINTER
   TYPE VARIABLES IN HIGH LEVEL LANGUAGES. */

GLOBVARS=≤:     <≠LP≠,LP>,                    /* LINE POINTER */
                <≠CS≠,CS>,                    /* CHANGE STRING */
                <≠LS≠,LS>,                    /* LOCATE STRING */
                <≠MS≠,MS>,                    /* MASTER STRING */
                <≠SC≠,HOWSAFE>,               /* SAFETY COUNT */
                <≠LINES≠,LINES>,              /* FILE LENGTH */
                <≠DATE≠,DATE>,                /* TODAYS DATE */
                <≠SYSTEM≠,SYSTNAME>,          /* THE SYSTEM NAME */
                <≠INPUT≠,TTYNAME>,            /* INFILE */
                <≠OUTPUT≠,TTYOUT>,            /* OUTFILE */
                <≠FILENAME≠,FILENAME>,        /* THE EDITFILE */
                <≠READNAME≠,READNAME>,        /* FILE BEING READ */
                <≠SUSNAME≠,SUSNAME>,          /* SUSPEND FILE */
                <≠LISTFILE≠,LISTFILE> ≥;      /* THE PAGE FILE */
IF VAL(ARG1) EQ. ≠ALL≠ THEN ALLFLAG = T.; END IF VAL;
X = IF TYPE(ARG1)EQ.≠NULL≠ THEN 0M.ELSE ABBR(VAL(ARG1),GLOBVARS);
(∨Z → GLOBVARS↑ALLFLAG OR.X EQ. 0M. OR. X EQ. Z ) .
PRINT. X; IF X NE. Z THEN PRINT. GLOBVARS(X);; END ∨Z;
GLOBVARS = NULT.;RETURN; END DISPLAY;
```

≠DUPL≠ IS A ROUTINE WHICH WILL DUPLICATE A SECTION OF THE FILE AT ANOTHER LOCATION, WHILE NOT TOUCHING THE ORIGINAL COPY OF IT. MOVE CALLS IT WITH THE DELFLAG SET TO TRUE SO THAT THE ORIGINAL LINES CAN BE DELETED WITHOUT LATER RECALCULATING THE POINTERS.

```
/*    DUPL L3              MEANS DUPL LP,L3,LP
      DUPL L6 L8           MEANS DUPL L6 L8 LP
      DUPL L3 L6 L7        MEANS DUPL L3 THROUGH L6 AFTER L7 */
DEFINE DUPL;
/* LET DELARG CALCULATE M1 AND M2 */
IFEMPTY;
OKFLAG = F.; IF NOT. DELARG(0) THEN RETURN; END IF NOT.;
M4 = DELFLAG; DELFLAG = F.;
M3=LOC(ARG3,LINES); IF M3 EQ. 0 THEN RETURN;;
/* WE NOW HAVE DUPL M1,M2,M3 --ALL VALID LINE NUMBERS */
RANGE = M2+1-M1; NCHANGES = NCHANGES + RANGE;
TABLE = TABLE(1:M3)+TABLE(M1:RANGE)+TABLE(M3+1:+TABLE);

IF M1 LT. M3 THEN RANGE = 0; END IF M1;
IF M4 THEN DEL(A+RANGE,B+RANGE);END IF M4;
GARBCOLL;  /* COMPRESS MEMORY AND REDUCE FIELD LENGTH */
OKFLAG = T.; RETURN;END DUPL;
```

≠DYNAMIC≠ SETS A FLAG WHICH DYNAMICALLY COPIES THE EDIT COMMANDS BEING GIVEN TO ANOTHER FILE, FOR LATER REUSE, ORIGINALLY, IT WAS INTENDED THAT SOMETHING LIKE THIS COULD GENERATE CDC TYPE UPDATE DIRECTIVES DYNAMICALLY, BUT THE PROBLEM PROVED UNSOLVABLE WITH THE RESOURCES AT HAND, SUCH AS A FINITE AMOUNT OF TIME, ETC. THE FILE TO WHICH THE DYNAMIC COPYING TAKES PLACE COULD LATER BE READ BY THE EDIT ≠READ≠ COMMAND, THUS REPRODUCING THE EFFECTS OF THE CURRENT INTERACTIVE EDITING SESSION IN A NON-INTERACTIVE ELVIRONMENT. BY THE WAY, EDIT AS IT WAS IMPLEMENTED AT NYU CAN BE EITHER INTERACTIVE OR USED IN ≠BATCH-MODE≠ WITH INPUT, SAY,FROM A CARD READER. THIS TAKES A GOOD THING AND MAKES IT BETTER.

```
DEFINE DYNAMIC; DYNAMING = T.; RETURN; END DYNAMIC;
```

THE EDIT COMMAND ≠ECHO≠ SETS THE ECHOING FLAG SO THAT ALL LOCATES AND CHANGES WILL PRINT OUT THE LINE LOCATED OR OR CHANGED TO.

```
DEFINE ECHO;SET (ECHOING);RETURN;END ECHO;
```

THE COMMAND EDIT INITIALIZES A FILE FOR EDITING, REQUESTING FROM THE USER PERMISSION TO SAVE AN UNSAVED FILE, IF THE USER HAD BEEN EDITING

ALREADY, AND IF A FILE NAME IS GIVEN, IT HASHES AND SETS UP A COPY OF THAT FILE. IF NO FILE NAME IS GIVEN IT JUST CLEARS WHAT IS CURRENTLY BEING EDITED.

```
DEFINE ZEDIT;
OKFLAG = EDIT(VAL(ARG1)); RETURN; END ZEDIT;
```

THE EDIT COMMAND ≠END≠ WILL ASK PERMISSION TO SAVE ANY UNSAVED FILE, AND THEN CLEAR MEMORY AND RETURN CONTROL TO THE MONITOR SYSTEM. EDIT IS NO LONGER IN CONTROL OF THE MACHINE. THIS IS THE ONLY CIVILIZED WAY TO LEAVE THE EDITOR. OTHER WAYS INCLUDE CPU OPERATOR INTERVENTION, HARDWARE ERRORS, AND SPIKES IN THE CPU POWER LINE.

```
DEFINE ENDEDIT;
WANTTOSAVE; CLEAROUT;  EDITING = F.;
EVICT(≠ZZZZZVE≠);  /* CLOSE AND UNLOAD THE EDIT SCRATCH FILE */
RETURN; END ENDEDIT;
```

THE EDIT COMMAND ≠ERROR≠ ABORTS ANY READ OR MACRO EXPANSION IN PROCESS AND CAN PRINT A MESSAGE. IT ALSO EVICT ALL COMMANDS IN THE STACK WAITING FOR DISPOSAL TO THE OPERATING SYSTEM FOR EXECUTION AND IGNORES ANY COMMANDS TYPED IN ON THE REST OF LINE ON WHICH THE ERROR COMMAND APPEARS.

```
DEFINE ZERROR; PRINT. VAL(ARG1); I = MSTOP; I = HALTREAD;
BACKSPAC;
/* SET FLAGS TO STOP READ OR MACRO */
OKFLAG = F.; RETURN; END ZERROR;
```

≠ESCAPE≠ IS A COMMAND WHICH RESETS THE ESCAPE CHARACTER. IT IS INCLUDED IN THE REPERTOIRE OF COMMANDS TO TAKE CARE OF THE HIGHLY UNLIKELY EVENT THAT A USER OF THE EDITOR HAS A DOCTORS NOTE INDICATING A BONA FIDE ALLERGY TO THE PARTICULAR CHARACTER ASSUMED BY THE SYSTEM TO BE THE DEFAULT ESCAPE CHARACTER. IT WOULD BE NEEDLESSLY CONFUSING TO A LATER USER TO HAVE TO WADE THROUGH A MACRO FILE SET UP WITH THE WRONG ESCAPE CHARACTER. PERHAPS THIS COMMAND SHOULD NOT BE IMPLEMENTED SINCE THE PURPOSES OF THE EDITOR DO NOT INCLUDE COMPLETE FREEDOM TO DYNAMICALLY MODIFY THE EDITOR TO THE POINT OF UNREADABILITY AND CONFUSION.

```
DEFINE ESCAPE;IF TYPE(ARG1)EQ.≠STRING≠ THEN
    CHESCAPE=(VAL(ARG1))(1:1);;
RETURN;END ESCAPE;


DEFINE ZEXECUTE;
IFEMPTY;
IF NOT. EXING AND. NOT. INFIT AND. DELARG(0) THEN EXING =T.;
    TRANSMIT. ≠NOEXECUTE≠;
    (~M2>=LP>=M1) TRANSMIT. GET(LP); END ~;
ELSE OKFLAG = F.;
END IF;
RETURN; END ZEXECUTE;
```

≠HALTREAD≠ STOPS READING A READ FROM A READ FILE AND RETURNS THE EDITOR TO READING FROM FILE WHICH IS THE NEXT STACKED FILE ON THE READSTACK.

```
DEFINE HALTREAD;      /* STOP READING FROM AN ALTERNATE FILE */
ALTRDING=F.;RECLINE=0M.;MACSTAK = NULT.;INPUTFILE = GET(INPUTSTACK);
RETURN; END HALTREAD;


DEFINE IFEQ;
IF ARG1 NE. ARG2 THEN DOSKIP(ARG3);;RETURN;END IFEQ;


DEFINE IFGE;
IF ARG1 LEXLT. ARG2 THEN DOSKIP(ARG3);;RETURN;END IFGE;


DEFINE IFGT;
IF ARG2 LEXLT. ARG1 THEN DOSKIP(ARG3);;RETURN;END IFGT;


DEFINE IFLE;
IF NOT. ARG1 LEXLT. ARG2 THEN DOSKIP(ARG3);;RETURN;END IFLE;


DEFINE IFLT;
IF NOT. ARG2 LEXLT. ARG1 THEN DOSKIP(ARG3);;RETURN;END IFLT;


DEFINE IFNE;
IF ARG1 EQ. ARG2 THEN DOSKIP(ARG3);;RETURN;END IFNE;


DEFINE IFTYPE;
IF VAL(ARG1) NE. TYPE(ARG2) THEN DOSKIP(ARG3);;RETURN;END IFTYPE;
```

```
DEFINE INSERT;
INSERTED = T.;  /* TO PREVENT INFINITE ITERATION */
IF TYPE(ARG1) EQ. ≠NULL≠ THEN DOINSERT; RETURN; END IF;
I=1;
(WHILE TYPE(ARG(I)) NE. ≠NULL≠ DOING I=I+1;)
    IF TYPE(ARG(I)) EQ. ≠INT≠ THEN PUTFOR. DEC. VAL(ARG(I));
    ELSE IF TYPE(ARG(I)) EQ. ≠NAME≠ THEN PUTFILE. VAL(ARG(I));
    ELSE Z=ARG(I); CHECK(Z,≠STRING≠); Z = VAL(Z);
        IF Z(1) EQ. CHESCAPE THEN PRINT. Z; READ. Z; ARG(I)=ZZZZZ;
        ELSE PUT(IMODE(Z)); END IF Z;
    END IF TYPE;
    END WHILE;
RETURN; END INSERT;


DEFINE LEXCLASS;
IF TYPE(ARG1) EQ. ≠STRING≠ AND. TYPE(ARG2) EQ.≠STRING≠
    AND. ↓(VAL(ARG1) IS. P1) EQ. 1 THEN
    LEXCLASS(P1) = VAL(ARG2);;
RETURN; END LEXCLASS;


DEFINE LIST; Z = VAL(ARG1); (↗1<=J<=↓Z) Z(J) IN. LISTSET;;
RETURN; END LIST;


DEFINE LOCATE;
IFEMPTY;
ANCHRING = ANCHORQ; ANCHORING = F.;
IF TYPE(ARG2) EQ. ≠NAME≠ THEN ILLARG; RETURN;;
X = LOC(ARG1,IF TYPE(ARG2) EQ. ≠NULL≠ THEN LINES ELSE LOC(ARG2));
/* NOT FOUND */
IF X EQ. 0 THEN ERROR(10);OKFLAG=F.; ELSE LP = X; DOECHO;;
ANCHORQ = ANCHRING;
RETURN; END LOCATE;


DEFINE LZONE;
IF ZONEARG THEN LZONE1=1;LZONE2=M2;;
RETURN; END LZONE;


DEFINE MACRO;
DOMACRO(TL. ARG1,(↑2<=J<=↓ARG)<TL. ARG(J)>);
RETURN; END MACRO;


DEFINE MARGIN;
```

```
RETURN; END MARGIN;


DEFINE MASTER;
IF TYPE(ARG1) EQ. ≠NULL≠ THEN MASTLINE = MS;
ELSE IF TYP(ARG1) EQ. ≠STRING≠ THEN MASTLINE = VAL(ARG1);
    ELSE ILLARG; RETURN;;
IF TYPE(ARG2) EQ.≠INT≠ AND. TYPE(ARG3) EQ.≠INT≠ THEN
    MZONE1 = VAL(ARG2); MZONE2 = VAL(ARG3);
ELSE IF TYPE(ARG2) EQ.≠NULL≠ THEN DOINSERT(MASTLINE);ELSE ILLARG;;
RETURN;END MASTER;


DEFINE MOVE;IFEMPTY; DELFLAG = T.;DUPL;RETURN;END MOVE;


DEFINE MSTOP;
MACRING = F.; Z = SKIPPING; SKIPPING = F.; OKFLAG= NOT.Z;
RETURN; END MSTOP;


DEFINE MZONE;
IF ZONEARG THEN MZONE1=M1;MZONE2=M2;; RETURN; END MZONE;


DEFINE NEXT;
IFEMPTY;
IF TYPE(ARG1) EQ. ≠NULL≠ THEN VAL(ARG1) = 0;
ELSE IF TYPE(ARG1) EQ. ≠INT≠ THEN LP = LINES MIN.(LP+VAL(ARG1))MAX.1;
ELSE ILLARG;; RETURN;END NEXT;


          ≠NOCC≠ CANCELS OUT THE CCING FLAG

DEFINE NOCC; CCING = F.;RETURN ;END NOCC;


          ≠NODYNAMIC≠ CALCELS OUT THE DYNAMIC FLAG

DEFINE NODYNAMIC; DYNAMING = F.;RETURN ;END NODYNAMIC;


          ≠NOECHO≠ CANCELS OUT THE ECHO FLAG

DEFINE NOECHO;(≠NOECHO≠)ILL.DANGERS;ECHOING = F.;
RETURN;END NOECHO;


DEFINE NOEXECUTE; EXING = F.; RETURN; END NOEXECUTE;
```

```
                    ≠NOLIST≠ CANCELS OUT SOME LISTING OPTION

DEFINE NOLIST;Z=VAL(ARG1));(∨1<=J<=+Z)Z(J) OUT. LISTSET;;
RETURN;END NOLIST;


                    ≠NONUMBER≠ CANCELS OUT THE NUMBERQ FLAG

DEFINE NONUMBER;NUMBERQ = F.;
IF TYPE(ARG2) NE. ≠NULL≠ THEN CCING = T.;;
RETURN;END NONUMBER;


DEFINE NOPAGE;PAGING =F.;RETURN;END NOPAGE;
/* FLUSH THE BUFFERS, WRITE THE EOF MARKS, AND
EVICT THE FILE */


DEFINE NOPROMPT; PROMPTQ = F.; RETURN; END NOPROMPT;


DEFINE NOUPD; UPDING = F.;  RETURN;END NOUPD;


          ≠NUMBER≠ SETS THE NUMBERQ FLAG INDICATING THAT
          WHEN A LINE IS PRINTED AT THE TELETYPE, THE NUMBER
          OF THE LINE SHOULD ALSO BE PRINTED.

DEFINE NUMBER; NUMBERQ = T.; RETURN; END NUMBER;


DEFINE OVERLAY;
IFEMPTY;
IF TYPE(ARG1) EQ. ≠NULL≠ THEN Z = NULC.;
    (WHILE Z(1:2) NE. (CHRSCAPE+(≠ ≠))DOING LP=LP+1;)
        READ. Z; DOOVRLAY(Z);END WHILE; RETURN;
   ELSE IF TYPE(ARG1) EQ. ≠STRING≠ AND. DFLARG(1) THEN
     (∨M1<=LP<=M2) DOOVRLAY(VAL(ARG1));END ∨M1;
   ELSE OKFLAG=F.; END IF;
RETURN; END OVERLAY;


DEFINE PAGE;
M1 = ≠OUTPUT≠; M3 = (≠ ≠); M2 = 60; J = 1; /* SET DEFAULTS */
(WHILE J LE. 3 AND. TYPE(ARG(J)IS.Z) NE. ≠NULL≠ DOING J=J+1;)
    IF TYPE( Z) EQ.≠NAME≠THEN M1 = VAL(Z);
    ELSE IF TYPE(Z) EQ.≠STRING≠ THEN M2 = VAL(Z);
    ELSE IF TYPE(Z)EQ.≠INT≠ THEN M3 = VAL(Z); ELSE OKFLAG=F.;;
    END WHILE;
```

```
/* LPLINES IS THE NUMBER OF LINES PER PRINTER PAGE */
PAGING = T.;LISTFILE = M1; LPLINES = M2;TTYOUT=LISTFILE;
PRINT, M3;   /* THE PAGE TITLE */
RETURN; END PAGE;


DEFINE PRINT;
IF DELARG(0) THEN
(~M1<=LP<M2) PRLINE(LP);;;
RETURN;END PRINT;


DEFINE PROMPT; PROMPTQ=T.; RETURN; END PROMPT;


DEFINE PZONE;
IF ZONEARG THEN PZONE1=M1;PZONE2=M2;; RETURN; END PZONE;


DEFINE READ;
READFILE = VAL(ARG1);
IF NOT, VALID. READFILE THEN OKFLAG=F.; RETURN;;
IF VAL(ARG2) EQ. OM. THEN REWINDF(READFILE); END IF;
PUT(READFILE,INPUTSTACK);
MACSTAK=≠COMMENT/TERMINATING≠+READFILE+≠/≠;

                    CANCEL ALL COMMANDS ON THE REST OF THIS LINE. THE
                    NYU EDITOR DOES THIS, PROBABLY FOR EFFICIENCY BUT
                    TO REMOVE THIS  UNDESIRABLE EFFECT  FROM THE SETL
                    EDITOR, JUST REMOVE THE FOLLOWING CARD.
RECLINE=OM.;
PRINT, (≠READING≠)+READFILE;
RETURN;END READ;


          ≠REMOVE≠ REMOVES MACRO DEFINITIONS FROM THE MACRO TABLE AND
          THE MACRO FILE.  IF THERE ARE NO MORE MACROS, THE RANDOM
          FILE CALLED ≠MACRO≠ IS EVICTED.

DEFINE REMOVE;
(WHILE MACROS(VAL(ARG(N))) NE. OM.) MACROS LESS,VAL(ARG(N));;
IF #MACROS EQ. 0 THEN EVICT. MACROFILE;;
RETURN; END REMOVE;


DEFINE REPLACE;
IFEMPTY; DELETE;
IF OKFLAG THEN LP=1 MAX. LP-1; /* CANT GET TO TOP EXCEPT EXPLICITLY*/
/* ARGS 3...N BECOME ARGS 1...(N-2) FOR THE INSERTION */
ARG = ARG(3:#ARG); INSERT; END IF; RETURN; END REPLACE;
```

```
DEFINE RESUME;
/* RESTART A SUSPENDED-FILE (CHECKPOINT RESTARTING) */
IF VALID. ARG1 THEN SUSNAME = VAL(ARG1);;
WANTTOSAVE;REWINDF(FILE);
IF READSETL(FILE) NE. ≠SUSPENDED≠ THEN
    PRINT. ≠FILE UNFIT FOR RESUME≠; OKFLAG=F.;
ELSE <STATUS,TABLE,MACROS,TEST> = READSETL(FILE);
OKFLAG=TEST EQ. ≠RESUME OK≠; END IF; RETURN;  END RESUME;


DEFINE SAFETY;ITSSAFE = T.;
HOWSAFE = IF TYPE(ARG1) NE. ≠INT≠ THEN 20 ELSE VAL(ARG1);
RETURN;END SAFETY;


DEFINE SAVE;
IFEMPTY;
IF VALID. ARG1 AND. DELARG(1) THEN SAVEIT(VAL(ARG1),M1,M2);
ELSE OKFLAG = F.;;
RETURN; END SAVE;


DEFINE SCALE;
PRINT. DEC. LP+≠>≠+ SCALER(PZONE1,PZONE2+1-PZONE1);
RETURN;END SCALE;


DEFINE SETVAR;
IF TYPE(ARG1) EQ.(≠STRING≠) AND. TYPE(ARG2) EQ. ≠STRING≠
    AND. +(VAL(ARG1) IS. X) EQ. 1
THEN GLOBVARS(X) = VAL(ARG2);
ELSE OKFLAG = F.;
END IF;
RETURN;END SETVAR;


                ≠SHOW≠PRINTS  A MACRO  WHOSE NAME WAS GIVEN AS THE
                ARGUMENT, AND IF NOT GIVEN,IT PRINTS JUST THE NAME
                OF ALL THE MACROS IN THE MACRO TABLE.

DEFINE SHOW;
IF TYPE(ARG1) NE. (≠STRING≠) THEN
     (∨MACRO → MACROS)PRINT. MACRO(1); END ∨ MACRO;
ELSE Y = ≤VAL(ARG(K)). 1<=K<=NUARGS≥;
    (∨Z→Y) IF (MACRO(Z) IS. X) NE. ON. THEN PRINT.Z;PRINT.X;
         ELSE PRINT.≠MACRO NAME NOT RECOGNIZED≠; END IF;
     END ∨;
END IF; RETURN; END SHOW;
```

```
DEFINE SKIP;
                    THIS  COMMAND WILL  SKIP ALL INPUT  LINES UNTIL IT
                    COMES TO  A LINE CONTAINING A  LABEL WHICH MATCHES
                    ITS ARGUMENT IF READING FROM AN ALTERNATE FILE.

IF EDITING THEN PRINT.≠CANT ENTER EDIT FROM EDIT≠;
ABORTFLAG=T.;RETURN;;
IF TYPE(ARG1) EQ. ≠STRING≠ THEN SKIPPER = VAL(ARG1); SKIPPING = T.;
ELSE ILLARG;; RETURN; END SKIP;


                    SPLIT  OFF A LINE  ON A COLUMN RETURNS TO PREVIOUS
                    INPUT FILE.OR A SUBSTRING OF THE LINE.
DEFINE SPLIT;
IFEMPTY;
PUSH(ECHO); NOECHO;
IF TYPE(ARG1) EQ. ≠INT≠ THEN Z = GET(LS); J = VAL(ARG1);
    PUT. Z(I: J MAX.150); LS = LS+I;PUT. Z((J+I)MAX.150:↓7);
ELSE IF TYPE(ARG1) EQ.≠STRING≠ AND.(LOC(ARG1,0)IS.ABORTFLAG)NE.0 THEN
    DOSPLIT; ELSE ILLARG; END IF;
POP(ECHO); RETURN; END SPLIT;


DEFINE STATUS;
STASET = IF(ABBR(VAL(ARG1),STANAMES) IS.X) NE. OM.
         THEN ≤;X≥
         ELSE ≤;1,2,3,4,5,6≥;
STALINE = NULT.;
STALINE(1) = LINEONE;
         STALINE(2) = ≠EC.DL,CC,TC,TABS, ≠+CHESCAPE
                                +CHBLANK+CHCONCAT+CHTAB+(≠ ≠)+TABS;
STALINE(3) = ≠ZONE,MZONE,PZONE,SAFETY≠+ZONE1+ZONE2+MZONE1+MZONE2+
         PZONE1+PZONE2+SAFETY;
STALINE(4) = LS;
STALINE(5) = CS;
STALINE(6) = MS;
STALINE(7) = ≤<X,LEXCLASS(X)>.X→ALPHS LESS.ABSOR+LEXCLASS(X) NE.OM.≥;
(↓1<=K<=↓STASET) PRINT. STALINE(K); END ↓;
RETURN;END STATUS;


                   TAKE A CHECKPOINT DUMP OF THE WHOLE EDIT
                   ENVIRONMENT WHICH WILL LATER BE RELOADED.
DEFINE SUSPEND;
IFEMPTY;
SUSNAME = VAL(ARG1); OPENF(2);
PRINT. ≠SUSPENDING≠ + SUSNAME;
WRITSETL(FILE,≠SUSPENDED≠);
```

```
WRITSETL(FILE,<STATUS,TABLE,MACROS,DANGERS,≠RESUME OK≠>);
NCHANGES = 0;RETURN;END SUSPEND;


DEFINE SYSTEM; J = GUESS(VAL(ARG1),WHO);
IF J EQ. 0M. THEN PRINT. ≠ILLEGAL SYSTEM≠; RETURN;;
X = SYSTABS(J IS. SYSTNAME); CHTAB = HD. X; TABS = TL. X;
IF TYPE(ARG2) EQ. ≠INT≠ THEN
    TL. X = GETTABS; SYSTABS(J) = <CHTAB,TABS>;END IF;
RETURN; END SYSTEM;


DEFINE TIMER;PRINT. TIME;RETURN;END TIMER;
```

TOP IS THE ONLY WAY TO POSITION THE LINE POINTER (LP) BEFORE THE FIRST LINE IN THE FILE FOR INSERTING BEFORE LINE 1. IT IS REDUNDANT IN THAT ANY SUCH INSERTION COULD FOLLOW LINE 1 AND BE FOLLOWED BY THE COMMAND MOVE L1 IT IS INCLUDED AS A USER-CONVENIENCE ONLY.

```
DEFINE TOP;  LP = 0;RETURN; END TOP;


DEFINE UNSAFE; SAFE = F.;HOWSAFE = 20; RETURN; END UNSAFE;


DEFINE UPD;   UPDING = T.;RETURN;END UPD;


DEFINE ZONE; /* ≠LZONE≠ AND ≠PZONE≠ TOGETHER */
IF ZONEARG THEN LZONE1=M1; LZONE2=M2; PZONE1=M1; PZONE2=M2;;
RETURN; END ZONE;
```

```
/* 6.2  EDIT SUBROUTINES  */
```

THE FUNCTION ASK ASKS A QUESTION WHICH REQUIRES A YES OR NO FOR AN ANSWER. THE PROGRAM ACCEPTS THE TYPED REPLY AND RETURNS T, IF THE ANSWER WAS YES AND F, IF THE ANSWER WAS NO. IF THE ANSWER WAS ANYTHING ELSE IT REPEATS THE QUESTION AD INFINITUM.

```
DEFINEF ASK, QUESTION;REPLY = NULC.;
(WHILE REPLY NE. ≠YES≠ AND. REPLY NE.≠NO≠)
    PRINT. QUESTION; REPLY=READSETL(TTYNAME);END WHILE;
RETURN REPLY EQ. ≠YES≠;END ASK.;
```

CAT IS A SIMPLE WAY TO EXAMINE AN OUTLINE OF THE FILE BEING EDITED. CAT SEARCHES THROUGH THE HASH TABLE FOR ALL SPECIAL LINES AND PRINTS OUT A TABLE OF THESE LINES, AND THE NUMBER OF LINES BETWEEN THEM. A TYPICAL TABLE MIGHT LOOK AS FOLLOWS:

```
42 LINES 43> >>>>>FILE
0  LINES 44> >>>>>0
```

THIS TABLE INDICATES THAT THE FILE CONTAINS 42 CARDS FOLLOWED BY A FILE INSERTION LINE (SEE SECTION ON INSERTIONS) WHICH IS FOLLOWED BY AN END-OF-FILE MARK AND THAT THERE ARE NO OTHER SPECIAL LINES IN THE FILE. WHEN EDITING VERY LARGE FILES WHICH CONTAIN MANY END OF FILE CARDS AND INSERTION LINES, THIS CAN BE A USEFUL UTILITY TO POSSESS. IT CAN ALSO BE USED LIKE THE SAVE COMMAND TO CAT ONLY A RANGE BETWEEN TWO GIVEN LINES.

```
DEFINE CAT;
IF NOT. DELARG(0) THEN OKFLAG = F.; RETURN ;;
IF M1 EQ. M2 THEN M1=1; M2=LINES;;
PREV = M1;
(WM1<=K<=M2 * SPECIAL(K))
    PRINT. DEC.(M1-PREV),≠LINES≠,DEC. K, GET(K);
    PREV=K;
    END W;
RETURN; END CAT;


DEFINE DOECHO;
IF ECHOING THEN PRLINE; END IF ECHOING;
RETURN; END DOECHO;
```

```
                    ENCIPHER MAPS EACH CHARACTER IN A STRING INTO A
                    UNIQUE CHARACTER WHICH IS DETERMINED BY THE
                    MAPPING CONTAINED IN THE GLOBAL VARIABLE NAMED AS
                    ARG1, BUT IF THE CHARACTER IS NOT IN THE DOMAIN OF
                    ARG1,IT IS MAPPED INTO ITSELF.
DEFINE ENCIPHER;
CODEMAP = GLOBVARS(VAL(ARG1));
PUSH(CHBLANK);CHBLANK = 0M.;
(=LZONE1 <=K<=LZONE?)
    CODEMAP(CURRENT(K)) ORM. CURRENT(K);;
DOCVRLAY(CURRENT); POP(CHBLANK);
RETURN; END ENCIPHER;


                    DOSKIP WILL EITHER SKIP AN INTEGRAL NUMBER OF
                    COMMAND LINES ON THE INPUT FILE OR IT WILL SKIP TO
                    A COMMAND LABELLED BY A CHARACTER STRING.

DEFINE DOSKIP(PAIR);
IF TYPE(PAIR) EQ. ≠STRING≠ THEN SKIPPING = T.;SKIPPER=VAL(PAIR);
ELSE SKIPPY(IF TYPE(PAIR)EQ. ≠INT≠ THEN VAL(PAIR)ELSE 1); END IF;
RETURN;END DOSKIP;


                    EDIT PREPARES A FILE FOR EDITING BY PREPARING A
                    HASH TABLE FOR THE LINES OF TEXT, DETERMINING THE
                    SYSTEM, AND SETTING THE INITIALIZATION PARAMETERS.
DEFINEF EDIT(FILENAME);
WANTTOSAVE; EVICT(≠ZZZZZVE≠);OPENF(≠ZZZZZVE≠);INITIAL;
IF NOT. VALID. FILENAME THEN FILENAME = ≠********≠; RETURN T.;;
REWINDF(FILENAME);
PUT(READSETL(FILENAME)IS.X);SYSTNAME=WHO(GUESS(X,SYSGUESS));
PRINT. ≠EDITING≠.FILENAME;
(WHILE NOT. EOFYET DOING LP = LP + 1) PUT(READSETL(FILENAME));;
LINEONE;
EDITING = T.; NCHANGES = 0; LP=0;
RETURN T.; END EDIT;


DEFINE ERRORS(K); PRINT. ERR(K); RETURN; END ERRORS;


ERR = <
    ≠ILLEGAL ARGUMENT TYPE≠,                                    /* 1 */
    ≠WRONG ORDER≠,                                              /* 2 */
    ≠ARG TO DELARG NOT ZERO OR ONE≠,                            /* 3 */
    ≠PATTERN SPECIFIER TOO LONG≠,                               /* 4 */
    ≠UNDEFINED GLOBAL VARIABLE≠,                                /* 5 */
```

```
    ≠REPEAT STRING TOO LONG≠,                                    /* 6 */
    ≠ILLEGAL CHARACTER IN CHANGE STRING COLON BRACKETS≠,          /* 7 */
    ≠GLOBAL VARIABLE TOO LONG≠,                                  /* 8 */
    ≠EDIT FILE OVERFLOW≠,                                        /* 9 */
    ≠NOT FOUND≠,                                                 /* 10*/
    ≠TAB IN REPEAT LOCATE≠,                                      /* 11*/
    ≠EDIT FILE EMPTY≠>;                                          /* 12*/


DEFINEF GETTABS;
K=2; X=NL.; (WHILE TYPE(ARG(K)) EQ.≠INT≠ DOING K=K+1;)ARG(K)IN. X;;
IF TYPE(ARG(K)) EQ. (≠STRING≠) THEN CHTAB =RG(K,1);;
RETURN X; END GETTABS;
```

GUESS ATTEMPTS TO MATCH ONE OF ITS SYSTEM PATTERNS AGAINST LINE 1 OF THE FILE.

```
DEFINEF GUESS(Z,GUESSES);
RETURN IF Ξ1<=K<=↓GUESSES↑MATCH(Z,GUESSES(K))THEN K ELSE 0;
END GUESS;


DEFINE IFEMPTY;
IF LINES EQ. 0 THEN ERRORS(12); OKFLAG = F;;
RETURN; END IFEMPTY;


/* INTRO EDIT IS CALLED THE FIRST TIME EDIT IS USED */
DEFINE INTROEDIT;PRINT. ≠EDIT VERSION BEGB1073≠;RETURN;END INTROEDIT;


DEFINEF LINEONE;
RETURN FILENAME + ≠;≠ + DEC. LP + ≠,≠ + DEC. LINES + ≠,≠ +
          DEC. NCHANGES    + ≠.≠+
          IF NOT. ECHOING THEN ≠NOECHO,≠ ELSE NULC. +
          IF NOT. NUMBERD THEN ≠NONUMBER,≠ ELSE NULC. +
          IF CCING THEN ≠CC,≠  ELSE NULC. +
          IF DYNAIMING THEN ≠DYNAMING,≠ ELSE NULC. +
          IF PAGING THEN ≠PAGED,≠ ELSE NULC. +
          IF NOT. PROMPTO THEN ≠NOPROMPT,≠ ELSE NULC. +
          IF NOT. SAFE THEN ≠UNSAFE,≠  ELSE NULC. +
          IF UPDING THEN ≠UPDING≠  ELSE NULC. +
          SYSNAME;
END LINEONE;
```

IN ANY INTERACTIVE ENVIRONMENT, THERE WILL BE A MAXIMUM NUMBER OF CHARACTERS THE SOFTWARE IN THE MULTIPLEX UNIT CAN TRANSMIT AT A TIME. IT IS

```
                    ASSUMED THAT THE  SETL PRINT  ROUTINE
                    AUTOMATICALLY   BREAKS   UP    LONG   LINES   INTO
                    TRANSMITTABLE SEGMENTS AND GENERATES THE NECESSARY
                    MULTIPLE SUB-WRITES.

DEFINE PRLINE; /* PRINT OUT CURRENT LINE */
GET(LP);
PRINT. (IF NUMBERQ THEN DEC. LP + ≠> ≠ ELSE NULC.)
    + CURRENT(PZONE1:PZONE2+1-PZONE1);
RETURN; END PRLINE;

                        /* DECIDE IF LINE INTERVAL IS TOO BIG TO DO */
DEFINEF SAFE(J,K);
IF NOT. ITSSAFE OR. ((K+1-J) IS. Z)  LE. HOWSAFE THEN RETURN T.;;
RETURN ASK. (DEC.Z+(≠ LINES SAFE≥≠)); END SAFE;


DEFINE SAVEIT(FILE);
OPENF(FILE); PRINT, ≠SAVING≠,FILE,LINES,NCHANGES,DANGERS;
PUSH(ITSSAFE); ITSSAFE = F.;
REWIND(FNT(SAVEFILE)IS.Z);
IF M1 EQ. M2 THEN M1 = 1; M2 = LINES; END IF M1;
(∨ M1 <=LP<= M2)  X=GET(LP);
    IF X EQ. EOR THEN WRITSETL(Z,ER);
    ELSE IF X(1) EQ. INSLINE THEN
        REWINDF(X(2:↓X) IS. X); J = NULC.;
        /* MERELY COPY THE FILE TO BE INSERTED */
        PRINT. (≠INSERTING≠)+X;
        (WHILE J NE. ER AND. J NE. OR.)
            J = READF(X); WRITSETL(X,J); END WHILE;
    ELSE WRITSETL(Z,X);
    END IF X;
    END ∨M1;
 POP(ITSSAFE);REWINDF(Z);LP = 1;NCHANGES =0;FILENAME=FILE;
RETURN; END SAVEIT;


DEFINE SKIPPY(INT);
RECLINE = OM.; J = ↓MACSTAK+1-INT;
IF J GT. 0 THEN MACSTAK=MACSTAK(1:J);RETURN;;
MACSTAK=NULT.;
(∨1<=J<INT) READ.Z;
    IF Z EQ. ER THEN INPUTNAME = GET(INPUTSTAK);OKFLAG=F.;RETURN;;
    END ∨;
RETURN;END SKIPPY;


DEFINEF TABIN(CARD);
IF (↓TABS) EQ. 0 THEN RETURN CARD;;
```

```
COL=1;CARD=(≠ ≠)+CARD+(≠ ≠));
(WHILE COL LE. ↓CARD DOING COL=COL+1;)
    IF CARD(COL) EQ. CHTAB THEN CARD=CARD(1:COL-1) +
    ((NEXTTAB(COL)+1-COL)*(≠ ≠)) + CARD(COL+1:↓CARD); END IF;
    END WHILE;
RETURN TRUNC. CARD(2:LZONE2 MIN. ↓CARD));END TABIN;


/* DECIDE IF THERE IS ENOUGH TIME LEFT TO DO COMMANDS */
DEFINEF TIMELEFT; RETURN T.; END TIMELEFT;



DEFINE WANTTOSAVE;
IF NCHANGES EQ. 0 THEN RETURN;;
IF ASK.≠WANT TO SAVE≠ THEN
    NAME = NULC.;
    (WHILE NOT. VALID. NAME) PRINT. ≠FILENAME>≠; READ. NAME;;
    SAVEIT(NAME);
ELSE PRINT. CURRFILE. ≠NO LONGER BEING EDITED≠; END IF ASK.;
RETURN; END WANTTOSAVE;
```

/* 7. EDIT ADJUNCTS */

ASIDENT -- A POSTPROCESSOR TO EDIT

THIS SECTION ASSUMES A KNOWLEDGE OF THE CDC UPDATE 2.0 PROGRAM.

ASIDENT INPUTS A 90 COLUMN UPDATE COMPILE FILE AND THE SAVE FILE AFTER THE COMPILE FILE WAS EDITED. ASIDENT COMPARES THE IDENTIFIERS IN THE UPDATE ID,SEQ FIELD AND OUTPUTS A SERIES OF UPDATE DIRECTIVES WHICH WILL CHANGE THE PROGRAM LIBRARY SO THAT IT WILL NOW GENERATE THE EDITED RATHER THAN THE ORIGINAL COMPILE FILE. THESE ARE WRITTEN ON THE FILE IDENT. ASIDENT IS A SYSTEM COMMAND, THUS IT IS USED IN THIS FORMAT:

ASIDENT(OLD,NEW)

ASIDENT REWINDS OLD AND NEW BOTH BEFORE AND AFTER, ASIDENT OPENS A FILE CALLED IDENT WHICH IS ALSO REWOUND AFTERWARDS.

CARD = <CHARS, ID,SEQ>
CHARS = THE CARD TEXT OF LENGTH LCARD
ID = THE UPDATE IDENT OF LENGTH LID
SEQ = THE UPDATE SEQUENCE NUMBER LSQ

AN EXAMPLE:

FILE1	FILE2
<≠MN≠,≠A≠,≠01≠>	<≠MN≠,≠A≠,≠01≠>
<≠X$≠,≠B≠,02≠>	<≠=Y≠,≠D≠,≠03≠>
<≠=Y≠,≠B≠,≠03≠>	<≠JJ≠,≠ ≠,≠ ≠>
<≠CS≠,≠D≠,≠21≠>	<≠AB≠,≠ ≠,≠ ≠>
<≠LL≠,≠D≠,≠22≠>	<≠LL≠,≠D≠,≠22≠>

AFTER EXECUTING ASIDENT(FILE1,FILE2)
WITH <LCARD,LSEQ,LID> = <2,2,1>
THE FILE IDENT WOULD CONTAIN:

```
*NOABBR
*DELETE B,2,B,2
*INSERT B,3
```

```
JJ
AB
*DELETE D.21,D.22
*ABBREV


DEFINE ASIDENT(OLD,NEW);
       CONTROL = (≠*≠);
       NOAB    = CONTROL + (≠NOABBR≠);
       AB      = CONTROL + (≠ABBR≠);
       DEL     = CONTROL + (≠DELETE≠);
       INS     = CONTROL + (≠INSERT≠);
       BEF     = CONTROL + (≠BEFORE≠);
       LSEQ    = 2;
       LCARD   = 2;
       LID     = 1;
       CONTROL = (≠*≠);
       PER     = (≠.≠);
       COMMA   = (≠,≠);
       BLNK    = (≠ ≠);

FILE1=MAKFILE(NEW,90);FILE2=MAKFILE(OLD,90);
REWIND(FILE1); REWIND(FILE2);
OUTFILE = MAKFILE(≠IDENT≠,72); /* FOR WRITE STATEMENTS */
PRINT, NOAB;
EOFYET = F.; A =NULL.; B = NULL.; J = 1;

    (WHILE NOT. EOFYET) J=J+1;
        (WHILE NOT. EOFYET AND. (TL. A) EQ. TL. B)
             PREVA=A;PREVB=B;
             A=SCAN(READSETL(FILE1)) ;B=SCAN(READSETL(FILE2));
             END WHILE NOT.;

        IF B(2) EQ. (LID*BLNK) THEN
             PRINT. IF PREV EQ. NULL.
                  THEN BEF + PREVA(2) + PER + PREVA(3)
                  ELSE INS  +  PREVB(2) + PER + PREVA(3);
             (WHILE NOT. EOFYET AND. B(2) EQ. (LID*BLNK))
                  B=READSETL(FILE2);PRINT. B;
                  IF B EQ. BB THEN EOFYET = T.;;
                  END WHILE NOT. EOFYET;
        ELSE  /* TL. A NE. TL. B OR ELSE ASSUMPTIONS VIOLATED */
             XX = DEL + A(2) + PER + A(3)+COMMA;
             (WHILE NOT. EOFYET AND. (TL. A) NE. TL. B )
                  X = A(2) + PER + A(3);
                  A = READSETL(FILE1);
                  IF A EQ. BB THEN EOFYET = T.;;
                  END WHILE NOT.;
             PRINT. (XX+X);
```

```
            X = NULT.; XX = NULT.;
        END IF;
    END WHILE NOT. EOFYET;

    IF A EQ. ER THEN
         PRINT. INS + A(2) + PER + A(3); PRINT. B;
         (WHILE B NE. ER)
              PRINT. READSETL(FILE2) IS. B;
              END WHILE B NE.;
    ELSE X = DEL + A(2) + PER + A(3) + COMMA;
         (WHILE B NE. ER)
              PREV = A; A = READSETL(FILE1);
              END WHILE B;
    END IF;

    PRINT. (X+PREV(2)+PER+PREV(3));
    PRINT. AB;
    <NOAB,AB,DEL,INS,BEF> = NULT.;
    <A,B,X,XX,PREV> = NULT.;
    REWIND(FILE1);REWIND(FILE2);REWIND(OUTFILE);
    RETURN; END ASIDENT;

COMPUTE; DO;
DEFINEF SCAN(CARD); RETURN
<CARD(1:LCARD),CARD(LCARD+1:LID),CARD(LID+1:LSEQ)>; END SCAN;

COMPUTE;DO;
PRINT,NOAB,AB,DEL,INS,BEF;

COMPUTE; DO;
ASIDENT(5,8);
COMPUTE; FINISH;
```

8. BIBLIOGRAPHY

COMMENTS ON THE SETL DRAFT — UNKNOWN
4/71 SNL 21

COMPATIBLE TIME-SHARING SYSTEM A PROGRAMMERS GUIDE — F CRISMAN
7/65 MIT

CRITICAL COMMENT ON THE SETL DRAFT — PAT GOLDBERG
11/70 SNL 2

EDIT DOCUMENTATION — M RYAN
6/73 NYU

EDT REFERENCE MANUAL — UNIVAC VMOS
9/73 SPERRY RAND — EA-016-2-00

IMPLEMENTATION OF RECURSIVE ALGORITHMS IN NON-RECURSIVE LANGUAGES — K ABDALI
6/72 NYU

LATEST EDIT DOCUMENTATION — M BRENNER
10/73 NYU

THE LINEAR QUOTIENT HASH CODE — J BELL, C KAMAN
11/70 CACM

ON PROGRAMMING (VOL 1) — J SCHWARTZ
2/73 NYU

POINTERS AND VERY HIGH LEVEL LANGUAGES — M MINSKY
1/73 SNL 96

PROGRAM TEXT EDITOR
8/72 CDC — 60307100

SCATTER STORAGE TECHNIQUES — R MORRIS
1/68 CACM

SET COMPARISON USING HASHING TECHNIQUE — M HARRISON
6/70 NYU — NYO-1480-155

SETL DATA BASE EXTENSIONS — G WINEBERGER
UNPUBLISHED

SETLB SPECIFICATION OF UPDATE — M BRENNER
5/73 NYU

SRTL--THE SETL RUN-TIME LIBRARY H WARREN
6/73 NYU

THEORY AND IMPLEMENTATION OF PATTERN MATCHING D MCENTYRE
10/72 NYU

A SHORT GUIDE TO <EDIT> AS IMPLEMENTED ON THE 6600 COMPUTER
FOR THE SCOPE 3.2, 3.3, AND 3.4 OPERATING SYTEMS

<EDIT> IS A UTILITY FOR CREATING AND MODIFYING FILES CONSISTING OF LINES OF TEXT; EG. PROGRAMS, DATA, DOCUMENTATION. TO START <EDIT> ON THE 6600, LOG IN ON A TIME SHARING REMOTE TERMINAL (INTERCOM) AND TYPE IN THE FOLLOWING:

```
EDIT.
```

THE RESPONSE WILL BE: (GIVING EDIT VERSION AND O.S. VERSION)

```
EDIT(VER ZZZZZ)  SCOPE 3.X
*******,0,0,0,ABSOLUTE
E=>
```

THE FIRST LINE IS EXPLAINED BELOW (SEE STATUS). ≠E=>≠ IS A PROMPT THAT <EDIT> GIVES WHENEVER IT IS READY FOR A COMMAND. COMMAND FORMAT IS

```
COMMAND,ARG1,ARG2,...,ARGK
```

NOTE PARTICULARLY THAT THERE IS NO PERIOD AT THE END OF AN <EDIT> COMMAND, A COMMAND TERMINATED BY A PERIOD IS ASSUMED TO BE FOR SCOPE OR INTERCOM AND IS PASSED ON TO THE SYSTEM FOR EXECUTION. (EDIT IS A ≠TRANSPARENT≠ UTILITY; SCOPE AND INTERCOM COMMANDS CAN BE EXECUTED DIRECTLY THROUGH EDIT. THIS INCLUDES EXECUTION OF PROGRAMS PREPARED BY THE USER.) MORE THAN ONE COMMAND MAY BE ENTERED ON ONE LINE BY SEPARATING THE COMMANDS WITH SEMICOLONS; EG.,

```
SAVE MYFILE;FTN(I=MYFILE,L=MYLIST);BATCH,MYLIST,PRINT,END.
```

<EDIT> ASSIGNS LINE NUMBERS TO THE LINES OF THE FILE BEING EDITED; LINE 1 IS THE FIRST LINE, LINE 2 THE SECOND, ETC. THESE NUMBERS ARE NOT PART OF THE LINE AND CHANGE AS INSERTIONS AND DELETIONS ARE MADE. THERE IS A LINE POINTER (≠LP≠ BELOW) WHICH DETERMINES WHERE INSERTIONS GO AND CAN BE USED TO LOCATE LINES. LINES MAY BE SPECIFIED EITHER BY CONTENT OR BY POSITION AND EITHER RELATIVE TO THE LINE POINTER OR ABSOLUTELY. LINE SPECIFICATION IS EXPLAINED UNDER THE LOCATE COMMAND; THE DESCRIPTION THERE APPLIES TO ANY COMMAND WHICH TAKES LINE SPECIFIERS AS ARGUMENTS.

A FILE IS OBTAINED FOR EDITING WITH THE COMMAND ≠EDIT, FILE.≠ WHILE A FILE IS BEING EDITED, CHANGES ARE NOT MADE ON THE FILE ITSELF BUT ON A SCRATCH COPY MADE BY EDIT. THE USER SAVES DESIRED EDITED VERSIONS WITH THE COMMAND ≠SAVE≠. SHOULD THE USER FAIL TO SAVE THE LATEST VERSION OF A FILE BEFORE STARTING TO EDIT A NEW FILE OR ENDING <EDIT>, <EDIT> WILL ASK ≠WANT TO SAVE≥≠ IF THE USER TYPES ≠YES≠, <EDIT> WILL ASK FOR THE FILE NAME FOR THE SAVE. AFTER SAVING, <EDIT> WILL PROCESS THE COMMAND WHICH PROMPTED THE QUESTION.

IN THE DESCRIPTIONS BELOW ≠K≠INDICATES A POSITIVE INTEGER,≠LS≠A LINE SPECIFIER (EXPLAINED UNDER LOCATE),≠LP≠THE LINE POINTER,AND ≠LFN≠THE NAME OF A LOCAL FILE.(IF NO LOCAL FILE OF THE NAME GIVEN EXISTS,<EDIT>WILL CREATE ONE.)COMMANDS MAY BE ABBREVIATED UP TO (AND IN SOME CASES BEYOND)AMBIGUITY BY DELETING ALL LETTERS AFTER THE N+1 TH SUCH THAT THE FIRST N DO NOT MATCH ANY OTHER EDIT COMMAND. THE SHORTEST ABBREVIATIONS ARE GIVEN IN THE FIRST COLUMN.

I. EDITING COMMANDS

ED	EDIT, LFN	SET UP FILE LFN FOR EDITING. LP SET TO 0
SA	SAVE, LFN	REWRITE LFN WITH CURRENT VERSION OF FILE BEING EDITED. IF LFN IS OMITTED, THE LFN MENTIONED IN THE MOST RECENT ≠SAVE≠ OR ≠EDIT≠ IS USED.
	SAVE, LFN,LS1,LS2	AS ABOVE, EXCEPT THAT ONLY THE LINES BETWEEN THOSE SPECIFIED BY LS1 AND LS2 (INCLUSIVE) ARE WRITTEN TOO LFN.
T	TOP	SET LP TO 0; INSERTIONS HERE GO BEFORE LINE 1
B	BOTTOM	SET LP TO LAST LINE OF FILE
N	NEXT, K NEXT,-K	SET LP TO LP+K. IF K IS OMITTED, 1 IS USED. SET LP TO LP-K (NOTE THAT THE COMMA IS OPTIONAL BETWEEN ANY COMMAND AND ARGUMENT WHEN THE ARGUMENT BEGINS WITH + OR -.)
L	LOCATE, LS	SET LP TO THE LINE SPECIFIED BY LS AND PRINT THE LINE. LS MAY BE EITHER RELATIVE TO LP OR ABSOLUTE.

- ABSOLUTE LINE SPECIFIERS -

T	LINE 0
B	LAST LINE OF FILE (BOTOM)
LK	LINE K (THE LETTER L FOLLOWED BY A DECIMAL INTEGER)

- RELATIVE LINE SPECIFIERS -

K	LP+K-1; K-TH LINE DOWN FROM LP COUNTING THE CURRENT LINE.
+K	LP+K
-K	LP-K
DSTRINGD	THE FIRST LINE FROM THE CURRENT LINE DOWNWARD WHICH CONTAINS THE STRING GIVEN. ≠D≠ MUST BE A DELIMITER AND THE BRACKETING DELIMITERS MUST BE IDENTICAL. THE DELIMITERS ARE / = ¬ < [>] ≠ ≥ ¶ = → ≤ THE UP ARROW AND THE EXCLAMATION POINT.
-DSTRINGD	THE FIRST LINE FROM THE CURRENT LINE UPWARD WHICH CONTAINS THE STRING GIVEN.
S	THE FIRST SPECIAL LINE FROM THE CURRENT LINE DOWNWARD. (SEE (2) AND (3) UNDER ≠INSERT≠ FOR DEFINITION OF SPECIAL LINES.)

-S THE FIRST SPECIAL LINE FROM THE CURRENT LINE UPWARD.

THE ARGUMENT LS MAY ALSO BE FORMED BY SIMPLE COMBINATIONS OF THE ABOVE.

- EXAMPLES -

L/VAR1/ FIND THE FIRST OCCURANCE OF VAR1 BELOW LP. NOTE THAT THE COMMA IS OPTIONAL BETWEEN ANY COMMAND AND ARGUMENT IF THE ARGUMENT BEGINS WITH A DELIMETER.

L, L220+/VAR1/ FIND THE FIRST OCCURANCE OF VAR1 FROM LINE 220 DOWN

L, B-≠X/Y≠ FIND THE LAST OCCURANCE OF X/Y IN THE FILE. NOTE THAT / COULD NOT BE USED AS A DELIMITER HERE

L/TION Z/+/DATA/ THIS MIGHT LOCATE THE FIRST DATA STATEMENT IN FUNCTION ZETA, IF LP IS APPROPRIATELY POSITIONED.

A ANCHOR, LS

SAME AS LOCATE, EXCEPT THAT IF A DELIMITED STRING APPEARS IN THE ARGUMENT THE MATCH MUST OCCUR STARTING IN COLUMN 1 OF THE LINE (BUT SEE ≠LZONE≠)

C CHANGE, DSTRING1D,ESTRING2E

CHANGE STRING1 TO STRING2. IF STRING1 DOES NOT OCCUR IN THE CURRENT LINE THE FILE WILL BE SEARCHED DOWNWARD FOR THE

D DELETE

BEHAVES AS PRINT, EXCEPT THAT THE LINES ARE DELETED INSTEAD OF PRINTED. LP IS SET TO LINE FOLLOWING LAST LINE DELETED. NOTE THAT ≠DELETE L45≠DOES NOT DELETE LINE 45; IT DELETES FROM LP THROUGH LINE 45.

I INSERT, ARG1,...,ARGK

INSERT ARGUMENTS 1 THROUGH K AFTER THE CURRENT LINE (LP). AN ARGUMENT MAY BE:

(1) A DELIMITED STRING, WHICH IS INSERTED AS A LINE.

(2)A LOCAL FILE NAME (SAY ≠LFN≠), WHICH IS INSERTED AS THE SPECIAL LINE ≠>>>>>LFN≠. WHEN THE NEXT ≠SAVE≠IS MADE, THIS LINE IS REPLACED WITH ALL THE LINES IN THE FIRST RECORD OF THE FILE LFN AT THE TIME OF THE SAVE.

(3)AN INTEGER, K, WHICH IS INSERTED AS THE SPECIAL LINE ≠>>>>>K≠. WHEN THE NEXT ≠SAVE≠IS MADE, THIS LINE IS REPLACED BY AN END OF RECORD OF LEVEL

K.(LEVEL 15IS AN END OF FILE.)

INSERT ENTER INSERT MODE AFTER THE CURRENT LINE. IN THIS MODE THE PROMPT IS ≠K>≠WHERE K IS THE LINE NUMBER.EACH LINE TYPED IN IS ENTERED AS A NEW LINE IN THE FILE. NOTE THAT COMMANDS ENTERED IN THIS MODE WILL NOT BE SEEN AS COMMANDS BUT ONLY AS NEW LINES TO BE INSERTED. TO ESCAPE FROM THIS MODE, ENTER A LINE CONTAINING ONLY THE ESCAPE CHARACTER,≠~≠. THE ESCAPE CHARACTER CAN BE CHANGED.(SEE ≠ESCAPE≠.)

N.B. BOTH THE FORM WITH ARGUMENTS AND THE FORM WITHOUT LEAVE LP SET TO THE LAST LINE INSERTED.

P PRINT — PRINT THE CURRENT LINE.
PRINT, LS — PRINT FROM LP THROUGH LINE SPECIFIED BY LS, INCL.
PRINT, LS1,LS2 — PRINT FROM LINE SPECIFIED BY LS1 THROUGH LINE SPECIFIED BY LS2, INCL. IF LS2 IS A RELATIVE SPECIFIER, IT IS RELATIVE TO LS1.

N.B. IN ALL CASES LP IS SET TO LAST LINE PRINTED.

R REPLACE
IDENTICAL TO ≠DELETE≠, EXCEPT THAT AFTER THE LINES ARE DELETED THE USER IS PLACED IN INSERT MODE IN THE ≠HOLE≠MADE BY THE ≠DELETE≠.

SPL SPLIT, DSTRINGn
SPLITS STARTING AT THE COLUMN IN WHICH STRING BEGINS. IN BOTH CASES, LP POINTS TO THE SECOND LINE,NOT THE FIRST.

MO MOVE, LS1,LS2,LS3
TAKE THE LINES FROM LS1 THROUGH LS2, AND INSERT THEM AFTER THE LINE SPECIFIED BY LS3. IF LS3 IS ABSENT, IT IS ASSUMED TO BE LP. IF RELATIVE,LS2 IS RELATIVE TO LS1,LS3 TO LP.

DU DUPLICATE, LS1,LS2,LS3
LIKE ≠MOVE≠, EXCEPT THAT THE LINES BETWEEN LS1 AND LS2 ARE NOT DELETED FROM THEIR ORIGINAL PLACES.

II, EDITING MODE CONTROL

AN EDIT SESSION BEGINS IN CERTAIN DEFAULT MODES. COMMANDS PRESENTED IN THIS SECTION MODIFY THESE MODES. UNLIKE THE EDITING COMMANDS, WHICH WHEN EXECUTED PERFORM SOME ACTION WITH RESPECT TO THE FILE BEING EDITED, THE MODE COMMANDS MODIFY THE ENVIRONMENT IN WHICH EDITING TAKES PLACE. FOR EXAMPLE,THE PROGRAMMER MAY CHANGE THE FIELD WITHIN WHICH THE ≠LOCATE≠COMMAND ACTS. FURTHER, THESE CHANGES OF MODE PERSIST ACROSS THE EDITING OF NEW FILES.

ALL EDITING TAKES PLACE WITHIN SOME ≠EDITING SYSTEM≠. THERE ARE TWO KINDS OF SYSTEMS:TABULAR AND INTERPRETIVE.IN TABULAR SYSTEMS, LINES INSERTED ARE ENTERED AS THEY ARE TYPED UNLESS THE TAB CHARACTER IS IN THE LINE;IN INTERPRETIVE SYSTEMS LINES ARE FORMATED ACCORDING TO THEIR CONTENT. AT PRESENT ≠FORTRAN≠,≠SIMSCRIPT≠AND ≠ABSOLUTE≠ARE THE ONLY INTERPRETIVE SYSTEMS IMPLEMENTED.

WHENEVER ANY PARTICULAR TABULAR SYSTEM IS INVOKED, THE LAST SET VALUES OF THE TABS AND THE TAB CHARACTER FOR THAT SYSTEM ARE ESTABLISHED. SEVERAL SYSTEMS WITH DEFAULT TAB SETTINGS ARE PROVIDED. TWO SYSTEMS, CALLED ≠T1≠AND ≠T2≠, ARE PROVIDED WITHOUT PRESET TABS SO THAT THE PROGRAMMER NEED NOT ALTER THE DEFAULT SETTINGS OF THE PRESET SYSTEMS TO ESTABLISH HIS OWN TAB SETTINGS. THE PROGRAMMER IS OF COURSE FREE TO ALTER ANY TABULAR SYSTEM. THE COMMANDS ≠TABS≠AND ≠SYSTEM≠EXPLAINED BELOW ARE THE MEANS TO THIS END. THE NAMES AND DEFAULT TAB SETTINGS FOR THE TABULAR SYSTEMS ARE GIVEN IN THE TABLE BELOW.

SYSTEM	DEFAULT TAB SETTINGS	DEFAULT TAB CHARACTER
ALGOL	7,10,13,16,19	↓
BAL	NONE	↑
BAL^M	NONE	↑
COBOL	8,12,16,20,24	↑
COMPASS	11,21,35	↑
DAP	6,12,30,35,40,45,50,55,60,65	↑
JOVIAL	8,18,20,50,60	↑
MIX	12,17,22,30	↑
QUIDDL	8,12,16,20,24	↑
PL1	12,15,18,21,24,27,30	↓
SETLB	7,11,15,19,23,27,31,35	↑
SIMULA	7,10,13,16,19	↓
SNOBOL	NONE	↑
SYMPL	8,18,20,50,60	↑
T1	NONE	↑
T2	NONE	↑

DESCRIPTIONS OF THE MODE COMMANDS ARE GIVEN BEGINNING ON THE NEXT PAGE.

LZ LZONE, N1,N2

SETS THE ZONE IN WHICH (LOCATE≠AND ≠CHANGE≠ACT TO BE COLUMNS N1 THROUGH N2 INCLUSIVE. DEFAULT IS 1 THROUGH 72. THE MINUMUM AND MAXIMUM SETTINGS ARE 1 AND 150. IN ANY CHANGE WHICH MOVES CHARACTERS PAST COLUMN N2, THOSE CHARACTERS ARE ELIDED FROM THE COLUMN N2, THOSE CHARACTERS ARE ELIDED FROM THE MOVED. SIMILARLY, IN ANY CHANGE WHICH RESULTS IN DECREASING THE NUMBER OF CHARACTERS, BLANK FILL IS ADDED AT COLUMN N2, THE CHARACTERS BEYOND THAT POSITION NOT BEING MOVED. IF NO ARGUMENTS ARE GIVEN, THE DEFAULT VALUES OF 1 THROUGH 72 ARE REESTABLISHED.

PZ PZONE, N1,N2

SETS THE FIELD TO BE PRINTED WHENEVER A LINE IS DISPLAYED (AS WITH ≠PRINT≠,≠LOCATE≠OR ≠CHANGE≠)TO BE COLUMNS N1 THROUGH N2 INCLUSIVE. DEFAULT IS 1 THROUGH 72. THE MINUMUM AND MAXIMUM SETTINGS ARE 1 AND 150.

ZO ZONE, N1,N2

SETS BOTH LZONE AND PZONE TO THE VALUES GIVEN. IF NO ARGUMENTS ARE GIVEN, DEFAULTS ARE SET FOR BOTH,IS 1 THROUGH 150.

TA TABS, SYSNAME,N1,...,NK,DCD

INVOKES SYSTEM SYSNAME, SETS TABS IN COLUMNS N1,...., NK,AND CHANGES THE TAB CHARACTER TO C.(≠DCD≠INDICATES A DELIMITED CHARACTER.)FOR EXAMPLE,

TABS,T1,10,20,30,/→/

INVOKES SYSTEM T1, SETS TABS IN COLUMNS 10,20,AND 30, AND DESIGNATES →AS THE TAB CHARACTER. IN ANY INSERTED LINE THE PRESENCE OF THE CHARACTER ≠→≠WILL LOGICALLY ADVANCE THE COLUMN TO THE NEXT COLUMN IN WHICH A TAB IS SET. EG., I/A→B→→C/WILL RESULT IN A LINE WITH A IN COLUMN 1, B IN COLUMN RESULT IN A LINE WITH A IN COLUMN 1, B IN COLUMN, ARGUMENTS,SYSNAME,N1,...,NK,AND DCD MAY BE OMITTED WITH OUT PROVIDING COMMAS TO INDICATE THE OMISSION.THAT IS,THE FOLLOWING ARE ALL VALID:

TABS,COMPASS

TABS,10,20,30
TABS/→/

ANY ARGUMENT NOT MENTIONED IN THE COMMAND WILL BE LEFT UNCHANGED.THUS, FOR ≠TABS,10,20,30≠TABS WILL BE SET IN COLUMNS 10,20, AND 30,THE TAB CHARACTER BE SET IN COLUMNS 10,20, AND 30,THE TAB CHARACTER BE SET IN COLUMNS 10,20, AND 30,THE TAB CHARACTER COMPASS WITHOUT CHANGING ANY TABS OR THE TAB CHARACTER FOR THAT SYSTEM.

N.B. WHENEVER TAB SETTINGS ARE SPECIFIED (I.E., N1,...,NK), ALL PREVIOUS TABS IN THAT SYSTEM ARE CLEARED. IF NO TABS SETTINGS ARE SPECIFIED, NONE ARE CLEARED.

SY SYSTEM, SYSNAME,N1,...,NK,DCD

THIS COMMAND IS SIMPLY A SYNONYM FOR ≠TABS≠. THE REASON FOR HAVING A SYNONYM IS TO PROVIDE AN EASY TO REMEMBER COMMAND FOR INVOKING A SYSTEM WITHOUT TO REMEMBER COMMAND FOR INVOKING A SYSTEM WITHOUT CHARACTER;EG.,≠SYSTEM,FORTRAN≠,≠SYSTEM,COMPASS≠,A THOUGH THE COMMANDS ≠SYSTEM≠AND ≠TABS≠ARE COMPLETELY SYNONYMOUS,IT IS CONVENIENT TO DOCUMENT NON-TABULAR SYSTEMS HERE.

SYSTEM,FORTRAN

IN THIS SYSTEM LINES MAY BEGIN WITH AN ALPHABETIC OR NUMERIC CHARACTER. THE CHARACTER ≠+≠(FOR CONTINUATION CARDS)OR THE CHARACTER ≠*≠(FOR CONTINUATION CARDS)OR THE CHARACTER ≠*≠(FOR IGNORED. INTERPRETATION OF THE LINE FOR FORMATTING IS AS FOLLOWS.

(1) LINES BEGINNING WITH AN ALPHABETIC: THE LINE IS LEFT JUSTIFIED IN COLUMN 7.

(2) LINES BEGINING WITH A NUMERIC: THE INITIAL NUMERIC STRING IS RIGHT JUSTIFIED IN COLUMN 5, A BLANK IS PLACED IN COLUMN 6, AND THE FIRST NON-NUMERIC IS PLACED IN COLUMN 7 (IT MUST BE ALPHABETIC).

(3) LINES BEGINNING WITH +: THE + IS PLACED IN COLUMN 6 WITH THE REST OF THE LINE FOLLOWING IT AS TYPED.

(4) LINES BEGINNING WITH *: THE * IS PLACED IN

COLUMN 1 AND THE REST OF THE LINE ENTERED AS TYPED.

IF THE FIRST NON-BLANK IS ANY OTHER CHARACTER, THE PROMPT:

BAD SYNTAX - RETYPE>

APPEARS AT THE TTY AND AN ACCEPTABLE LINE MUST BE ENTERED.

EXAMPLE:

INSERT/PRINT 500//500FORMAT(3H HI)/

IS ENTERED AS

```
COL: 123456789
          PRINT 500
      500 FORMAT(3H HI)
```

THE SAME RESULT OF COURSE WOULD BE OBTAINED BY ENTERING THESE LINES IN INSERT MODE, MAKING NO ATTEMPT TO GET THE CHARACTERS INTO THE PROPER COLUMNS.

SYSTEM SIMSCRIPT

THIS IS IDENTICAL TO SYSTEM FORTRAN

SYSTEM ABSOLUTE

THIS IS THE INITIAL DEFAULT SYSTEM. IN THIS SYSTEM THERE IS NO TAB CHARACTER AND ALL LINES ARE ENTERED EXACTLY AS TYPED.

DE DEVICE, N

INFORMS EDIT OF THE CARRIAGE SIZE OF THE DEVICE FROM WHICH THE PROGRAMMER IS OPERATING. DEFAULT IS 72.

ES ESCAPE, DCD

SETS THE ESCAPE CHARACTER TO C.(≠DCD≠REPRESENTS A DELIMITED CHARACTER). DEFAULT IS ~. THIS CHARACTER DELIMITED CHARACTER). DEFAULT IS ~. THIS CHARACTER DEFINITION,OVERLAY,AND MASTER MODES.

EC ECHO
NOE NOECHO

DEFAULT IS ECHO MODE, IN WHICH CHANGE,LOCATE,AND INSERT WITH ARGUMENTS CAUSE THE LINE CHANGED,LOCATED,OR INSERTED TO BE DISPLAYED AT THE TELETYPE. THE COMMAND ≠NOECHO≠STOPS SUCH DISPLAY AND IS USEFUL WHEN A GREAT MANY SUCH LINES WOULD BE GENERATED, USUALLY BY A LARGE NUMBER OF CHANGES MADE BY ONE COMMAND.

NU NUMBER
NON NONUMBER
CC CC
NOC NOCC

DEFAULT IS NUMBER MODE, IN WHICH THE LINE NUMBERS ARE PRINTED TO THE LEFT OF DISPLAYED LINES. NONUMBER SUPPRESSES PRINTING OF THESE LINE NUMBERS. IF THE ARGUMENT ≠CC≠IS GIVEN WITH THE COMMAND ≠NONUMBER≠, THE FIRST CHARACTER OF EACH LINE IS INTERPRETED AS CARRIAGE CONTROL.

AS A CONVENIENCE THE COMMANDS ≠CC≠AND ≠NOCC≠ARE PROVIDED SO THAT A PROGRAMMER OPERATING IN NONUMBER MODE MAY SWITCH BACK AND FORTH BETWEEN CARRIAGE CONTROL MODE AND NO CARRIAGE CONTROL MODE. THESE TWO COMMANDS HAVE NO EFFECT IN NUMBER MODE.

PA PAGE, LFN SEND ALL EDIT OUTPUT TO FILE LFN, UNTIL A NOPAGE IS GIVEN.

PR PROMPT, NOPROMPT DURING MACRO, INSERT, OR MASTER MODES
GIVE THE PROMPT M>OR LP>WHERE LP IS THE DECIMAL VALUE OF THE LINE POINTER IF YOU ARE PROMPTING, IF NOT, THEN ACCEPT LINES

WHENEVER THE USER TYPES THEM, GIVING NO INDICATION THAT THEY HAVE BEEN ACCEPTED.

ST STATUS, ARG1,...., ARGK

THE ≠STATUS≠COMMAND PRINTS INFORMATION CONCERNING THE VARIOUS MODES WHICH HAVE BEEN SET. IF THE ONLY ARGUMENT TO ≠STATUS≠ IS ≠ALL≠(I.E.,≠STATUS,ALL≠)THE FOLLOWING TYPICALLY IS DISPLAYED.

```
EDITDOC,4388,5053,5,T1
EC=↓,TC=↑,BL=$,TABS=8,20,25
DEV=72,Z=1-80,PZ=1-80,MZ=11-72,PS=0,SC=25
L=/GRAT/
C=/GREAT/
M=*
U=
```

(THIS PARTICULAR EXAMPLE WAS OBTAINED BY TYPING ≠STATUS,ALL≠ WHILE CONSTRUCTING THIS DOCUMENT.)

LINE 1 SHOWS THAT THE CURRENT FILE NAME IS EDITDOC, LINE POINTER IS AT LINE 4388, THERE ARE 5053 LINES IN THE FILE, 5 CHANGES WERE MADE SINCE THE LAST SAVE, AND THE SYSTEM IS ≠T1≠. THIS IS THE ONLY LINE GIVEN IF STATUS IS GIVEN NO ARGUMENTS.

LINE 2 GIVES THE ESCAPE CHARACTER, THE BLANK CHARACTER (SEE SECTION VI), AND THE TAB SETTINGS FOR THE CURRENT SYSTEM.

LINE 3 GIVES THE DEVICE SIZE, THE LOCATE ZONE (Z) THE PRINT ZONE, THE MASTER ZONE(SEE SECTION VI), THE PAGE SIZE (SECTION VI),AND THE SAFETY COUNT.

LINES 4 AND 5 GIVE THE LOCATE STRING AND THE CHANGE STRING, RESPECTIVELY.

LINE 6 GIVES THE MASTER LINE. (SEE SECTION VI.)

LINE 7 GIVES THE USER DEFINED LEXICAL CLASS. (SEE SECTION 5)

UP UPDATE

IF THE UPDATE FLAG IS SET, COLUMNS 81 TO 90 ARE BLANKED OUT TO FACILITATE POSTPROCESSING BY THE ASIDENT ROUTINE. THIS IS BECAUSE IT IS IMPRACTICAL TO DYNAMICALLY GENERATE UPDATE DIRECTIVES CORRESPONDING TO THE EDIT COMMANDS ENTERED.

III. ITERATION CONTROL

THE USER MAY CAUSE A LINE OF COMMANDS TO BE ITERATED BY APPENDING AN ITERATION COUNT AS THE FINAL COMMAND ON THE LINE; E.G.

```
L/SUBROUTINE /;I≠       COMMON/MYBLOCK/MYVAR1,MYVAR2≠;5
```

THIS WILL CAUSE THE COMMON STATEMENT TO BE INSERTED AFTER EACH OF THE NEXT FIVE SUBROUTINE DECLARATIONS IN THE FILE BEING EDITED. INFINITE ITERATION IS INDICATED BY AN ASTERSIK; VIZ., TO PRINT THE FIVE LINES SURROUNDING EVERY ASSIGNMENT TO VARIABLE MYVAR3, ONE MAY WRITE

```
T
P/MYVAR3 =/-2,5;*
```

NOTE THAT THE COMMAND ≠T≠ WAS PLACED ON A SEPARATE LINE SINCE WE DO NOT WANT TO GO TO THE TOP AT EACH ITERATION.

THE LINE POINTER (LP) IS INCREMENTED AT EACH ITERATION. ITERATION IS TERMINATED IF A LOCATE FAILS, THE LINE POINTER IS AT THE OR BOTTOM WHEN THE ITERATION COMMAND IS ENCOUNTERED, OR THE ITERATION COUNT IS EXHAUSTED.

TO PREVENT INFINITE ITERATION AND UNDESIRABLE CONSEQUENCES CERTAIN HEURISTICS ARE USED TO TERMINATE ITERATION IMMEDIATELY. NO LINE CONTAINING A SYSTEMS COMMAND WILL EXECUTE THAT SYSTEMS COMMAND TWICE, THUS THE LINE

```
L/ABC/;REWIND FILE.;L/JKL/;2
```

WOULD LOCATE/ABC/, REWIND FILE, LOCATE/JKL/, INCREMENT THE LP, LOCATE/ABC/, AND THEN PROCEED TO THE NEXT LINE.

IV, THE <EDIT> MACRO FACILITY

A SEQUENCE OF COMMANDS MAY BE DEFINED AS A MACRO, SCOPE, INTERCOM AND <EDIT> COMMANDS MAY BE FREELY MIXED IN MACRO DEFINITIONS. UP TO 100 MACROS MAY BE DEFINED AT ANY GIVEN TIME. <EDIT> FEATURES WHICH ENHANCE THE POWER OF MACROS INCLUDE: LABELS ON COMMANDS; UNCONDITIONAL SKIP TO A LABEL; CONDITIONAL SKIP TO A LABEL UPON INPUT FROM THE PROGRAMMER AT EXECUTION; INTERACTIVE SUBSTITUTION OF ARGUMENTS TO EDIT COMMANDS DURING EXECUTION; READING OF COMMAND INPUT FROM PREPARED FILES; SUPENSION OF THE STATE OF <EDIT> AND THE EDIT FILE AND LATER RESUMPTION; COMMENTING TO PROVIDE INFORMATION TO THE PROGRAMMER DURING MACRO EXECUTION. MACROS MAY NOT CALL OTHER MACROS.

1. MACRO DEFINITION: THE COMMAND

```
MACRO,NAME,PARAM1,...,PARAMK
```

BEGINS MACRO DEFINITION. THE NAME AND FORMAL PARAMETERS MUST BE ALPHA NUMERIC AND NO MOR THAN 9 CHARACTERS EACH. THE RESPONSE TO THIS COMMAND WILL BE THE PROMPT ≠M>≠. A LINE OF COMMANDS ENTERED AFTER THIS PROMPT WILL BECOME PART OF THE MACRO DEFINITION AND WILL BE ANSWERED BY ANOTHER SUCH PROMPT. THE MACRO DEFINITION IS TERMINATED BY A LINE CONTAINING THE ESCAPE CHARACTER ≠↑≠ ONLY. (BUT SEE ≠ESCAPE≠) THE FORMAL PARAMETERS WILL BE RECOGNIZED AS SUCH WHEREVER THEY ARE NOT EMBEDDED IN ALPHA-NUMERIC STRINGS. TO EMBED A FORMAL PARAMETER IN SUCH A STRING, USE THE MACRO CATENATION SYMBOL, EXCLAMATION POINT. (TO INCLUDE THE ACTUAL CHARACTER EXCLAMATION POINT IN A MACRO DEFINITION, USE A DOUBLE EXCLAMATION POINT.) EXAMPLE:

```
E=>MACRO,DEBUG,N,VARIABLE,FORM
M>I/* START DEBUG//PRINT 99↓N,VARIABLE/
M>I/99↓N FORMAT(* AT N, VARIABLE IS *,FORM)/
M>I/* STOP DEBUG/
M>↑
```

UPON EXECUTION THE ARGUMENT SUBSTITUTED FOR N WILL BE CATENATED WITH 99 TO YEILD THE STATEMENT NUMBER BUT WILL NOT REPLACE THE N IN ≠PRINT≠. THE ARGUMENT SUBSTITUTED FOR WILL BE SUBSTITUTED AT THE END OF THE FORMAT STATEMENT BUT WILL NOT REPLACE THE FIRST FOUR LETTERS OF ≠FORMAT≠. L N

2. MACRO EXECUTION: A MACRO MAY BE EXECUTED AS THOUGH IT WERE AN <EDIT> COMMAND; VIZ.,

```
NAME,ARG1,...,ARGK
```

THE ARGUMENTS MAY BE OF ANY LENGTH AND ANY FORM ACCEPTED AS AN <EDIT> ARGUMENT; I. E., (1) DELIMITED STRANGS, (2) ALPHA-NUMERIC STRINGS, (3) PLUS, MINUS, OR ASTERISK, (4) CATENATION OF (1),(2) AND

(3) IN WHICH DELIMITED STRINGS ARE NOT ABUTTED. DELIMITERS ARE PASSED WITH THEIR THEIR STRINGS EXCEPT THAT WHEN AN ENTIRE ARGUMENT IS SURROUNDED WITH SINGLE QUOTES, THEY WILL BE STRIPPED UPON SUBSTITUTION. THIS PERMITS SUBSTITUTION OF ARGUMENTS WHICH THEMSELVES CONTAIN COMMAS, BLANKS, OR OTHER SEPARATORS. FOR NULL ARGUMENTS, THE NULL STRING IS SUBSTITUTED. CO

3, INTERACTIVE COMMMAND ARGUMENTS; DYNAMIC ARGEMENT SUBSTITUTION DURING MACRO EXECUTION IS OBTAINED BY PUTTING A PROMPT DELIMITED BY COMMERCIAL AT SIGNS IN PLACE OF THE ARGUMENT TO BE SUBSTITUTED DURING EXECUTION.Eg.,

CHANGE/THIS STRING/~WHAT DO YOU WANT≥~

WHEN THE SECOND ARGUMENT IS ENCOUNTERED DURING EXECUTION OF THIS COMMAND, ≠WHAT DO YOU WANT≥≠ IS PRINTED AT THE TTY. THE PROGRAMMER THEN ENTERS THE ARGUMENT TO REPLACE THE PROMPT. IN THIS EXAMPLE A DELIMITED STRING WOULD BE THE ONLY LEGAL TYPE-IN. THERE IS NO INTERACTIVE SUBSTITUTION OF ARGUMENT TO SCOPE OR INTERCOM COMMANDS. ONLY ARGUMENTS TO <EDIT> COMMANDS ARE EXAMINED FOR ~ DELIMITERS.

4, COMMAND FLOW CONTROL: ≠MSTOP≠,LABELS,≠SKIP≠,≠ASK≠

MSTOP - (ABBREV, MS)

WHEN ENCOUNTERED, THIS COMMAND CAUSES MACRO EXECUTION TO STOP.

LABELS

LABELS ARE ALPHA-NUMERIC STRINGS OF 9 OR FEWER CHARACTERS SEPARATED FROM THE COMMAND THEY LABEL BY A COLON; VIZ.,

THISLABEL:BATCH,FILE,PUNCH,END.

A COMMAND MAY BE GIVEN AS MANY LABELS AS DESIRED, A COLON SEPARATING EACH LABEL FROM THE FOLLOWING ONE. LABELD COMMANDS NEED NOT BE AT THE BEGINNING OF LINES.

SKIP,ARG - (ABBREV, SK)

ARG MAY BE EITHER AN UNSIGNED INTEGER OR A LABEL, IN THE FIRST CASE THE NUMBER OF COMMANDS SPECIFIED IS SKIPPED; IN THE SECOND COMMANDS ARE SKIPPED UNTIL A COMMAND WITH THE LABEL GIVEN IS ENCOUNTERED. IF ARG IS OMMITTED, IT IS TAKEN AS 1.

ASK,DSTRINGD,ARG

THIS COMMAND PRINTS ((≠STRING≠)) AS A PROMPT TO THE PROGRAMMER. THE PROGRAMMER MAY REPLY ≠YES≠ OR ≠NO≠. ≠YES≠ CAUSES THE ASK COMMAND TO

BEHAVE AS A NO-OP; ≠NO≠ CAUSES THE ASK COMMAND TO BEHAVE AS A SKIP, WITH ARG AS ABOVE. EXAMPLE:

ASK/SHALL I EXECUTE THE NEXT COMMAND≥/,SECTION2

UPON EXECUTION THE PROMPT ≠SHALL I EXECUTE THE NEXT COMMAND≥≠WILL BE PRINTED AT THE TELETYPE. IF THE PROGRAMMER REPLIES ≠YES≠, THE ASK COMMAND WILL ACT AS A NO-OP, PASSING CONTROL TO THE NEXT COMMAND; IF THE PROGRAMMER REPLIES ≠NO≠, COMMANDS WILL BE SKIPPED UNTIL THE LABEL ≠SECTION2≠IS ENCOUNTERED.

5. OTHER COMMANDS USEFUL IN CONJUNCTION WITH MACROS: ≠REMOVE≠, ≠SHOW≠ ≠READ≠, ≠SUSPEND≠, ≠RESUME≠, ≠COMMENT≠

ABBREV. COMMAND

REM REMOVE, NAME1,...,NAMEK

CAUSES THE DEFINITIONS OF MACROS NAMED TO BE REMOVED. IT IS NOT NECESSARY TO REMOVE MACROS BEFORE REDFINING THEM.

SH SHOW, NAME1,...,NAMEK

SHOWS THE DEFINITIONS OF THE MACROS NAMED. THE FORMAL PARAMETERS ARE REPRESENTED AT THE TTY BY ↑A,↑B,↑C,... WHERE ↑A REPRESENTS THE FIRST PARAMETER,↑B THE SECOND AND SO ON. IF NO ARGUMENTS ARE GIVEN, THE NAMES OF ALL DEFINED MACROS ARE LISTED.

REA READ, LFN

THIS COMMAND CAUSES <EDIT>TO READ ITS COMMAND INPUT FROM THE FILE LFN INSTEAD OF THE TELETYPE. CONTROL RETURNS TO THE TTY WHEN THE COMMAND ≠HALTREAD≠(ABBREV,≠HAL≠)OR AN END OF RECORD IS ENCOUNTERED.HENCE, IF MACRO DEFINITIONS ARE MAINTAINED ON A PERMANENT FILE,ONE NEED ONLY ATTACH THE FILE AND ≠READ≠IT TO CAUSE THE MACROS TO BE DEFINED.

TO INITIALLY CREATE A FILE OF MACROS, GIVE THE COMMAND ≠EDIT, MACROS≠AND IMMEDIATELY ENTER INSERT MODE. CREATE THE MACROS, ENDING EACH WITH THE ESCAPE CHARACTER AS THE ONLY CHARACTER ON THE LINE--OF COURSE, TO DO THIS WITHOUT LEAVING INSERT MODE,YOU MUST ENTER A DOUBLE ESCAPE CHARACTER.ENTER A SINGLE ESCAPE CHARACTER AFTER THE FINAL MACRO THUS LEAVING INSERT MODE AND SAVE AND CATALOG YOUR SAVED FILE.

ANY COMMANDS FOLLOWING A READ COMMAND ON THE SAME LINE WILL BE LOST. FURTHER, IF A MACRO INVOKES THE ≠READ≠COMMAND, THAT READ WILL NOT OCCUR UNTIL AFTER THE MACRO HAS FINISHED

EXECUTION. IF THE READ FILE INVOKES MACROS, THE MACROS WILL BE EXECUTED WHEN INVOKED. NOTE THAT IF THE READ FILE CONTAINS THE COMMAND ≠I≠(INSERT)WITHOUT ARGUMENTS, SUCCESSIVE LINES OF THE READ FILE WILL BE INSERTED UNTIL A LINE CONSISTING OF THE ESCAPE CHARACTER IS ENCOUNTERED. THEREAFTER LINES WILL AGAIN BE TAKEN AS COMMANDS.

SU SUSPEND, LFN

PRESERVES THE CURRENT STATE OF <EDIT>AND THE EDIT FILE BY TAKING A CHECKPOINT DUMP OF MEMORY AND THE EDIT SCRATCH FILE, ZZZZVE, IN A SPECIAL FORMAT ON LOCAL FILE LFN (WHICH IS THEREBY OVERWRITTEN). IF LFN IS ABSENT A DEFAULT NAME IS USED.

RES RESUME, LFN

RESTARTS EDITING FROM THE STATE PREVIOUSLY SUSPENDED ON LFN. IF LFN IS ABSENT THE DEFAULT FILE NAME USED BY SUSPEND IS USED.

CO COMMENT, DSTRINGD

CAUSES THE STRING BETWEEN THE DELIMITERS TO BE PRINTED AT THE TTY AS A COMMENT. IN COMPLEX MACROS WHICH INTERACT WITH THE PROGRAMMER SUCH COMMENTS MAY BE USED TO GUIDE OR INFORM THE PROGRAMMER DURING EXECUTION OF THE MACRO.

V. EDIT PATTERN MATCHING

SINCE EVERY PART OF EVERY LINE OF EACH EDIT FILE IS SO COMPLETELY HASH-CODED, THE PATTERN MATCHING CAN EASILY TAKE MUCH MORE COMPLICATED STRUCTURES, TENDING SOMEWHAT IN THE DIRECTION OF SNOBOL PATTERNS. IT IS POSSIBLE TO MATCH SUCH PATTERNS AS A SPAN OF BLANKS, EVERYTHING UP TO COLUMN 36, A SPAN OF NON-NUMERIC CHARACTERS, OR 3 ALPHABETIC CHARACTERS. YOU MAY USE THESE IN ANCHOR, LOCATE, OR CHANGE STATEMENTS.

THE PATTERN MNEMONICS ARE

A MATCHES ALPHARETICS
N MATCHES DIGITS
S MATCHES SPECIAL CHARACTERS (THAT IS ANYTHING BESIDES ANOB)
O MATCHES + - * /
B MATCHES BLANKS
U MATCHES THE USER DEFINED LEXICAL CLASS
↑ MATCHES EVERYTHING UP TO A COLUMN (UPWARD ARROW ON TTY)

THE RELATIONS ARE

=3 MEANING A LENGTH EXACTLY EQUAL TO 3
<7 MEANING OF LENGTH LESS THAN OR EQUAL TO 7 (NOT JUST LESS THAN)
>1 THE LENGTH IS AT LEAST 1

ALL OF THESE SPECIAL PATTERNS MATCHING A LEXICAL CLASS OF CHARACTERS CAN BE MIXED IN WITHIN SIMPLE STRINGS SEPARATED BY COLONS. THIS IS WHY YOU MUST USE A DOUBLE COLON IN A CHANGE STATEMENT WHERE YOU MEAN TO CHANGE A COLON TO A SEMICOLON FOR INSTANCE. THE FORM OF LEXICAL CLASS PATTERNS MATCHING IS PRESENTED THROUGH THE FOLLOWING EXAMPLES.

```
L/!(B)=16!/
```

THIS WILL LOCATE A STRING OF 16 BLANKS

```
C/$$!(B)=5:$//$$$/
```

THIS WILL CONDENSE OUT EXACTLY 5 BLANKS BETWEEN DOLLAR SIGNS
OR ELSE IT WILL FAIL.

```
C/!L(A)=2!//:L:99/
```

THIS LOCATES 2 ALPHABETIC CHARACTERS AND STORES THEM IN GLOBAL VARIABLE L IT CHANGES THEM TO THEMSELVES FOLLOWED BY 99. THERE CAN BE UP TO 10 GLOBAL VARIABLES IN USE AT ONE TIME.

```
C/!X(AN)=4:!Y(O)=1:!Z(AN)=4!//:Z:!Y:!X:/
```

THIS MATCHES 4 ALPHANUMERICS AND STORES IN TEMPORARY VAIRIABLE X THE VARIABLE X IS NOT USEABLE AFTER THE CHANGE STATEMENT. IMMEDIATELY FOLLOWING IS A SINGLE OPERATOR WHICH WILL BE STORED IN Y, FINALLY MATCH 4 OR MORE ALPHANUMERICS AND STORE IN Z. CHANGE THIS LOCATED STRING TO Z, Y, X. FOR EXAMPLE THIS WOULD CHANGE ITOP+JBOT INTO JBOT+ITOP.

```
L/:X↑7:Y(ANSO)>1::Z(B):=/
```

EVERYTHING UP TO BUT NOT INCLUDING COL 7 IS MATCHED AND STORED IN X

Y MATCHES AT LEAST ONE NON-BLANK, Z MATCHES AT LEAST ZERO BLANKS, THAT IS ALL BLANKS UP TO A AN EQUAL SIGN. IF DATA CARDS HAVE FIRST NAME AND LAST NAME BEGINNING IN
COL 1 AND ENDING IN COL 20, AND YOU WANT TO SWITCH THEM SO THE LAST NAME IS FIRST.

```
ZONE 1,20:C/:J(A)>1::(B)>1::K(A)>1::(B)>1://:K: :J:/
```

TO DEFINE A LEXICAL CLASS TO MATCH THE FORTRAN INTEGER LETTERS AND THEN FIND AN INTEGER ON THE RIGHT OF AN EQUAL SIGN.

```
LEXCLASS/IJKLMN/:L/=:(B)::(U)=1::(AN)<6::(SOB)>1:/
```

NOTE THAT EACH PART OF THE PATTERN DEFINED BY THE CLAS IS OPTIONAL, EVEN DOWN TO THE LEVEL OF LEAVING ALL PARTS OUT THAT IS TWO COLONS SEPARATED BY A BLANK SIGNIFIES A PATTERN THAT MATCHES ANYTHING AT ALL. (SNOBOL ARB) IT WILL ALWAYS TRY TO MATCH THE NULL STRING BEFORE ANYTHING LONGER.

```
L/IF: :THEN/    WILL FIND A ONE-LINE IF...THEN STATEMENT IN ALGOL.

C/PRINT:(B)::X(N)::(B):,//WRITE(2,:X:)/:*
```

WILL CHANGE FORTRAN2 TYPE PRINT STATEMENTS INTO FORTRAN-4 WRITE STATEMENTS FOR ANSI STANDARD PUBLICATION OF ALGORITHMS.

```
L/≠: :≠/  WILL LOCATE A PAIR OF MATCHED QUOTES ON A LINE

</*: :ABC: :*/<     WILL FIND A ONE-LINE PL/I-TYPE
                    COMMENT COTAINING THE STRING ≠ABC≠

L/(~: :>=/:*  WILL LOCATE ALL DESCENDING FOR-ALL LOOPS IN SETLB

SPLIT/ABC/:C/:↑1::(B)::X(ANSO)=1://:X:/
```

THIS WILL SPLIT OFF ABC AND ALIGN IT ON COLUMN 1 IN THE NEXT LINE THE :↑1: HAS THE EFFECT OF ANCHORING THE LOCATE ON COLUMN ONE. IN SEARCHING LOCATES WHERE THE LOCATED SUBSTRING IS EXPECTED MANY LINES AWAY, IT IS ALWAYS QUICKER TO FIND A HARD PATTERN AS THE FIRST

PATTERN IN THE LINE SINCE THE SCAN IS ALWAYS LEFT TO RIGHT. IT IS FASTER TO ANCHOR THE LOCATE PATTERN AND THEN CHANGE IN A SEPARATE COMMAND IF THE :↑1: IS FOLLOWED BY A HARD PATTERN. THUS

```
A/XXX/;C,,/YYY/
```

IS FASTER THAN

```
C/:↑1:XXX//YYY/
```

IF THE PATTERN HAS TO SEARCH BEYOND THE CURRENT CARD

A MORE GENERAL PURPOSE ALIGN COULD BE DESIGNED AS A MACRO ALN THIS WILL ALIGN THE CURRENT LINE ON COLUMN COL BY REMOVING ALL PRECEEDING BLANKS AND REPLACING THEM WITH COL BLANKS. THE FOLLOWING MACRO DOES SO FOR LINES A TO B

```
MACRO ALN COL LINEA LINEB
ZONE 1 150;NOECHO;C/:↑1::(B)::X(ASNSC)=1://:X:/,1,LINEB
LOCATE LINEA;C/// /,COL,LINEB
ECHO;PRINT;ZONE 1 80
```

THE FOLLOWING EDIT LINE WILL TRUCATE ALL 7-LETTER VARIABLE NAMES TO 6 ALPHANUMERICS.

```
TOP;C/:X(A)=1::Y(AN)=5::(AN)=1::Z(SOB)=1://:X::Y::Z:/,*,B;TOP =
```

TO CHANGE A SETL OPERATOR, SAY MIN. X, TO A SUBROUTINE, SAY MIN(X) FOR ALL INSTANCES IN A GIVEN SETL SUBPROGRAM, SAY XBAR,

```
C/ MIN,:(B)::X::Y(SOB)=1:// MIN(:X:) :Y:/,*,/END XBAR;/
```

TO CHANGE THE CENTRAL MEMORY PARAMETER ON A JOB CARD FROM ANYTHING, SAY CM45000, TO CM100000.

```
LEXCLASS/,,/;C/CM:(N)::X(U)=1://CM100000:X:/
```

LET US ASSUME THAT YOU HAVE SOME CARDS TO BE ADDED TO YOUR EDIT FILE IN DIFFERENT PLACES. YOU HAVE SET UP A DECK OF SAY SIX GROUPS OF INSERTION CARDS SEPARATED BY END-OF-RECORD MARKS (EDITOR SPECIAL LINES)ON FILE A, THE FOLLOWING MACRO WILL ALLOW YOU TO BREAK THE FILE APART INTO SIX INSERTION FILES AND INSERT THEM INTO THE CORRECT PLACE IN THE MAIN FILE,SAY FILE MAIN.

```
MACRO BREAKUP FILE
UNSAFE;EDIT FILE;
SAVE ZZZZZ1 TOP S; DELETE TOP S;
SAVE ZZZZZ2 T S; D T S
SAVE ZZZZZ3 T S; D T S
SAVE ZZZZZ4 T S; D T S
```

```
SAVE ZZZZZ5 T S: D T S
SAVE ZZZZZ6 T S: D T S
↓
MACRO PUTIN NUMBER AFTER
LOCATE AFTER; INSERT ZZZZZ7↓NUMBER
↓
BREAKUP A; EDIT MAIN
PUTIN 1 ≠/CHAPTER 2/-1≠
PUTIN 2 L342
PUTIN 3 2
PUTIN 4 B
PUTIN 5 L1
PUTIN 6 ≠END WHILE NOT. BLUE≠
SAVE NEWFILE
```

MACRO SUNDAY IS A DEVASTATINGLY NECESSARY MACRO TO PREVENT LOSS OF FILES UNDER ADVERSE DISK CONDITIONS. IT HAS THE EFFECT OF PERIODICALLY SUSPENDING THE EDIT FILE, CATALOGING IT (THE SUSPENDED FILE)AS A PERMANENT FILE,THAT IS,MAKING IT SYSTEM-FAILURE INDEPENDENT, AND PURGING AN OLD ONE AS YOU GO.THIS WASTES CPU TIME,REAL TIME,MASS STORAGE,AND TAXES THE ABILITY OF THE OPERATING SYSTEM TO HANDLE EXTREMELY HIGH LOADS OF INPUT/OUTPUT COMMANDS IN SORT PERIODS OF TIME,BUT ON THE PARTICULAR SUNDAY IT WAS INVENTED,IT WAS NECESSARY FOR SURVIVAL OF ANY FILES. IN PARTICULAR, BECAUSE OF VOLTAGE SPIKES IN THE POWER LINE, OR OTHER UNKNOWN FACTORS,THE OPERATING SYSTEM WAS FAILING EVERY FEW EDIT COMMANDS.

IN OUR PARTICULAR OPERATING SYSTEM, YOU CAN ONLY HAVE FIVE PERMANENT FILES OF A GIVEN FILE NAME,SO THIS MACRO TAKES TWO ARGUMENTS, THE NUMBER OF THE CYCLE OF THE EARLIEST FILE SUSPENDED ON THE FIFO STACK AND THE NUMBER OF THE LATEST PLUS ONE. IF YOU HAVE CURRENTLY NUMBERS 4,5,6,7, YOU WOULD GIVE AS THE PARAMETER 4 AND 8, AND THE MACRO WOULD SUSPEND YOU,CREATE CYCLE EIGHT FOR YOU,AND DESTROY CYCLE FOUR SO YOU HAVE ONE EMPTY SPACE TO CATALOG THE NEXT SUSPENSE. UNDER THE SCOPE 3.4 SYSTEM, THE NUMBERS ARE SUPERFLUOUS,SINCE YOU CAN SPECIFY LC=1 TO INDICATE YOU WANT THE LOWEST EXISTING CYCLE TO BE PURGED, AND SPECIFY NO CYCLE NUMBER AT ALL TO INDICATE THAT YOU WANT TO CATALOG ONE MORE THAN THE HIGHEST EXISTING CYCLE.

```
MACRO SUNDAY OLD NEW
RETURN ZZZZZVF BAD;SUSPEND
CATALOG(ZZZZZVF,ZZZZZVF,LID=SUS,CY=NEW,RP=999);
PURGE(BAD,ZZZZZVF,ID=SUS,CY=OLD)
↓
```

A PROMINENT AND RESPECTED MEMBER OF THE COMMUNITY GOT HOLD OF A BOARD OF ELECTIONS LISTING OF NAMES THE EDITOR WAS ASSIGNED TO THE TASK OF CONDENSING THE FILE AND REFORMATTING IT. THERE ARE NAMES ADDRESS, AND TELEPHONE NUMBERS IN THE FILE, ALONG WITH OTHER INFORMATION,SUCH AS AGE OF MOTHER,HOW MUCH TAX WAS PAID LAST YEAR, HOW TALL ARE YOU,ETC. HE WANTS THIS INFORMATION SPACED OUT, AND HE ONLY WANTS TO SEE THOSE WITH RC IN COLUMNS 61-62 TO INVITE THEM TO A BINGO GAME. THE EDITOR IS TO DELETE ALL OTHERS FROM THE FILE,AND SPREAD OUT THE NAMES AND ADDRESSES OF THE REMAINDER TO BE MORE EASILY READABLE. NAMES ARE IN COL 1-20;ADDRESSES 21-40;ZIP CODE 41-45;BDAY 51-56;REL 61-62;HEIGHT 67-69;(CM.)TAX CODE 70;TELEPHONE NUMBER 71-80.

```
MACRO RCLIST
NOECHO;ZONE 61 62; UNSAFE;NOLIST M
COMMENT/NEXT LINE DELETES ALL BUT THE FIRST 1000 DESIRED
I/$/;D, /RC/-1;N=1;1000
DELETE,B;SAFE;TOP;ZONE 1 80;COMMENT≠CONDENSED≠
LOCATE/;X=20:;Y=20:;Z=51:=51:B=61;67:;T=3:;H=1:;A=3;/
CHANGE,,/ :X: :Y: :7: :B: :T: :H:  :A:-/;*
COMMENT≠REFORMATTED≠
FILES,;ECHO;SAVE NUMB
```

THE FOLLOWING MACRO GETDECK IS AN EXAMPLE OF HOW EDIT CAN SAVE REAL TIME FOR ITS USERS. LET US SAY THAT FOR EFFICIENCY IN TIME AND SPACE AND FOR REDUCTION OF THE AMOUNT OF COMPUTER RESOURCES THAT YOU ARE USING UP,YOU HAVE PLACED ALL OF YOUR PROGRAMS ON A LARGE MASS STORAGE RESIDENT PROGRAM LIBRARY USING THE UPDATE PREPROCESSOR. WHEN YOU WANT TO EDIT AND EXECUTE ANY PARTICULAR PROGRAM, JUST INVOKE THIS MACRO,WHICH GETS THE DECK,AND EDITS IT,SO THAT YOU DONT HAVE TO TYPE IN SOME 12 ASSORTED COMMANDS EACH TIME. HUMANS MAKE FEWER MISTAKES WHEN THEY HAVE FEWER CHARACTERS TO TYPE.

```
MACRO GETDECK DECK FILE
RETURN ZZZZZG1,ZZZZ7G2;ATTACH(ZZZZZG2,FILE)
EDIT ZZZZZG1;I/*C DECK/;SAVE
UPDATE(Q,D,8,I=ZZZZ7G1,P=ZZZZ7G2,L=0)
RETURN ZZZZZG1,ZZZZ7G2
˅
```

THE FOLLOWING GROUP OF EDIT COMMANDS ARE USED TO LEXICALLY CHANGE SETLB SOURCE PROGRAMS INTO PUBLICATION SETL FOR PRINTING ON A PRINTER WHICH HAS THE COMPLETE 64-CHARACTER SET INCLUDING UNDERLINING.THEY HAVE A VERY SIMILAR EFFECT TO ABE GETZIERS SNOBOL4 PROGRAM, BUT AS A MATTER OF DECISION, THEY DO NOT DIVIDE THE PROGRAM INTO TWO COLUMNS, FIRST COLUMN FOR PROGRAM AND SECOND COLUMN FOR COMMENTS,SO THAT YOU CAN RIP AWAY THE COMMENTS.

```
T;C/$//$$/,*,B
T;C/≤;;//≤/,*,B
T;C/;;//$;:/,*,B
T;C/ LE.// $≤/,*,B;T;C/ GE.// $≥/,*,B
T;C/ LT.// </,*,B;T;C/ GT.// >/,*,B
T;C/ EQ.// $≡/,*,B;T;C/ NE./≠$≡;;/≠,*,B
T;C/<=//$≤/,*,B;;C/>=//$≥/,*,B
T;C/ AND.//  ^/,*,B;T;C/ OR.// $˅/,*,B
T;C/ OM.// $¬/,*,B;T;C/ NOT.// ¬/,*,B
T;C≠ NL,≠≠ 0:;/≠,*,B
MACRO PERIOD TOKEN LENGTH
C/TOKEN↓,//TOKEN;:→/
C/;:→//;;;:→→/;N-1;LENGTH
C/;:→///
˅
LEXCLASS/<>0≤≥/
MACRO ITER I J K
TOP
C/;;X>>1;;A(U)>1;:Y(ANB)>1;;B(U)>1;:Z(ANDB)=1;/
   /J;X;;A:K:Y:;B:;7:/;*
```

```
˅
C/ NOT,!X(ANUOB)>1!→/1 !X!$→!!/≠!*
C≠ NOT, Ξ≠≠Ξ!!/≠!*
T
C≠NOT, ~≠≠~!!/≠!*
ITER ≠ ~≠ ≠ ≠ ≠~≠
ITER ≠ Ξ≠ ≠ ≠≠ Ξ≠
ITER ≠(~≠ ≠(≠ ≠~≠
ITER ≠(Ξ≠ ≠(≠ ≠Ξ≠
COMMENT≠READY TO PLOT≠

T!C/$//$$/,*,B
T!C/≤!!//≤/,*,B
T!C/!!//$!!/,*,B
T!C/ LE.// $≤/,*,B;T!C/ GE.// $≥/,*,B
T!C/ LT.// </,*,B;T!C/ GT.// >/,*,B
T!C/ EQ.// $Ξ/,*,B;T!C/ NE./≠$Ξ!!/≠,*,B
T!C/<=//$≤/,*,B!!C/>=//$≥/,*,B
T!C/ AND,//  ~/,*,B!T!C/ OR.// $⌣/,*,B
T!C/ OM.// $¬/,*,B;T!C/ NOT.// ¬/,*,B
T!C≠ NL,≠≠ 0!!/≠,*,B
MACRO PERIOD TOKEN LENGTH
C/TOKEN↓,//TOKEN!!→/
C/!!→//!!!!→→/;N←1;LENGTH
C/!!→///
˅
LEXCLASS/<>0≤≥/
MACRO ITER I J K
TOP
C/I!X>>1!!A(U)>1!!Y(ANB)>1!!B(U)>1!!Z(ANDB)=1!/
   ^J!X!!A:K:Y!!B!!?!/!*
˅
C/ NOT,!X(ANUOB)>1!→/1 !X!$→!!/≠!*
C≠ NOT, Ξ≠≠Ξ!!/≠!*
T
C≠NOT, ~≠≠~!!/≠!*
ITER ≠ ~≠ ≠ ≠ ≠~≠
ITER ≠ Ξ≠ ≠ ≠≠ Ξ≠
ITER ≠(~≠ ≠(≠ ≠~≠
ITER ≠(Ξ≠ ≠(≠ ≠Ξ≠
COMMENT≠READY TO PLOT≠
```

SOME PROBLEMS WERE EXPERIENCED WITH THIS MACRO. A TABLE IN THE PLOTTING ROUTINE WHICH DRIVES THE CHARACTER PLOTTER HAD TO BE MODIFIED TO INCLUDE THE SETL SPECIAL CHARACTERS. THE PLOTTED CHARACTERS WERE TOO SQUARE SINCE THE TABLE WAS ACCURATE TO .01 INCHES. THE PLOTTING TIME WAS NOT A FACTOR SINCE GENERATING THE PLOT COMMANDS FOR LINEAR TEXT IS QUICK AND EASY AND THE PLOTTER AT NYU IS OFF-LINE AND IS USED LESS THAN TWO HOURS A DAY. THUS NO INTERFERENCE WITH THE NORMAL

OPERATION OF THE COMPUTER CENTER IS FORSEEN, EVEN IF LARGE AMOUNTS OF SETL TEXT IS PLOTTED.

VI, THE MASTER MODE

SOMEWHAT LIKE INSERT MODE IS THE MASTER MODE,BUT INSTEAD OF INSERTING THE LINE DIRECTLY, THE LINE IS OVERLAYED ON TOP OF A PRE-SET FRAME LINE, CALLED THE MASTER LINE. INITIALLY,THE MASTER LINE IS SET BY DEFAULT TO A SINGLE ASTERISK IN COLUMN 1 OF THE LINE, SO THAT WHEN INSERTING LINES UNDER MASTER MODE, THEY WILL HAVE THE APPEARANCE OF AN ASSEMBLY LANGUAGE COMMENT, WITH THE INFORMATION ENTERED FROM THE TELETYPE OVERLAYING THE COLUMNS SPECIFIED BY THE MASTER ZONES, (11 TO 72 BY DEFAULT),AND THE COMMENT CHARACTER IN COLUMN 1.

MZ MZONE

OF COURSE THERE IS A COMMAND ≠MZONE≠TO SET THE MASTER ZONES EXACTLY AS ZONE AND PZONE SET THE LOCATE AND PRINT ZONES RESPECTIVELY. THERE ARE TWO WAYS TO INVOKE THE MASTER COMMANDS.

MA MASTER

FIRSTLY, YOU MAY SAY JUST ≠MASTER≠,IN WHICH CASE YOU ENTER INSERT MODE, BUT ANY LINES ENTERED ARE OVERLAYED ON TOP OF THE MASTER STRING. TO LEAVE INSERT MODE, TYPE THE ESCAPE CHARACTER AS THE ONLY CHARACTER ON THE LINE. SPECIAL LINES ARE INSERTED UNCHANGED,OF COURSE.

SECONDLY, YOU MAY TYPE

≠MASTER,/A NEW MASTER STRING/≠

THIS ALSO PUTS YOU INTO INSERT MODE, BUT ALSO CHANGES THE SETTING OF THE MASTER LINE.AS AN EXAMPLE, LET US ASSUME THAT YOU WANT TO WRITE FORTRAN TYPE COMMENTS FROM COLUMN 11 TO 21 AND 41 TO 61 WITH 5 STARS IN COL 22 TO 26, AND A ≠C≠ IN COLUMN ONE. THE FOLLOWING SERIES OF COMMANDS WOULD SET UP SUCH AN INSERTION AND PUT YOU INTO INSERT MODE.

```
MZONE 11,61;TABS 41;MASTER=C                    $$$$$
```

THE FOLLOWING TABLE OF SPECIAL CHARACTERS AND THEIR TELETYPE KEYBOARD COUNTERPARTS MAY BE HELPFUL.

TELETYPE CHARACTER		APPEARS ABOVE	PRINTER CHARACTER	SETL CHARACTER
EXCLAMATION	1	↓	11-6-8	NUMBER SIGN
DOUBLE QUOTE	2	≡	0-6-8	THERE EXISTS
NUMBER SIGN	3	→	0-5-8	ELMT OF
DOLLAR SIGN	4	$	11-3-8	
PERCENT SIGN	5	¬	12-6-8	ARB
AMPERSAND	6	^	0-7-8	
SINGLE QUOTE	7	≠	4-8	
OPEN PARENTHESIS	8	(	0-4-8	
CLOSE PARENTHESIS	9	)	12-4-8	
COLON		:	2-8	
AT SIGN	P	∨	11-0	FOR ALL
BACKSLASH	L	≤	5-8	OPEN CURLY BRACKET
OPEN BRACKET	K	[	7-8	OPEN SQUARE BRACKET
CLOSE BRACKET	M	]	0-2-8	CLOSE SQUARE BRACKET
UP ARROW	N	↑	11-5-8	SUCH THAT
LESS THAN	,	<	12-0	
GREATER THAN	.	>	11-7-8	
QUESTION MARK	/	≥	12-5-8	CLOSE CURLY
LEFT ARROW MARK	0		NONE	(RUBOUT)

TELETYPE CHARACTER		APPEARS ABOVE	PRINTER CHARACTER	SETL CHARACTER
EXCLAMATION	1	↓	11-6-8	NUMBER SIGN
DOUBLE QUOTE	2	≡	0-6-8	THERE EXISTS
NUMBER SIGN	3	→	0-5-8	ELMT OF
DOLLAR SIGN	4	$	11-3-8	
PERCENT SIGN	5	¬	12-6-8	ARB
AMPERSAND	6	^	0-7-8	
SINGLE QUOTE	7	≠	4-8	
OPEN PARENTHESIS	8	(	0-4-8	
CLOSE PARENTHESIS	9	)	12-4-8	
COLON		:	2-8	
AT SIGN	P	∨	11-0	FOR ALL
BACKSLASH	L	≤	5-8	OPEN CURLY BRACKET
OPEN BRACKET	K	[	7-8	OPEN SQUARE BRACKET
CLOSE BRACKET	M	]	0-2-8	CLOSE SQUARE BRACKET
UP ARROW	N	↑	11-5-8	SUCH THAT
LESS THAN	,	<	12-0	
GREATER THAN	.	>	11-7-8	
QUESTION MARK	/	≥	12-5-8	CLOSE CURLY

LEFT ARROW MARK	0	NONE	(RUBOUT)

19,05.16,LP14 5.425 KIN.