SMEAR 2
A Pre-Processor for OS 360 JUL

Rock Denham
Automatic Data Analysis Group
CTEM # 32

April 1963
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>SMEAR 2 vs. SMEAR 1</td>
<td>1</td>
</tr>
<tr>
<td>Critical Events</td>
<td>1</td>
</tr>
<tr>
<td>Deck Set Up and Other Considerations</td>
<td>2</td>
</tr>
<tr>
<td>SMEAR Syntax</td>
<td>3</td>
</tr>
<tr>
<td>SMEAR Commands</td>
<td>3, 4</td>
</tr>
<tr>
<td>SMEAR Source Listing</td>
<td>10</td>
</tr>
<tr>
<td>SMEAR Improvements</td>
<td>11</td>
</tr>
</tbody>
</table>
Introduction:

IBM's OS 360 Job Control Language (JCL) is very powerful, very difficult to learn, and very difficult to use. It follows that if one provided some reasonable subset of OS 360 JCL's capabilities to the 360 programmer in a manner that was easy to learn, convenient to use, and relatively non error prone, his efforts would do much to calm the anxiety and ease the suffering of the great unwashed mass of JCL users.

SMEAR 2 is a modest attempt to attain this goal.

SMEAR 2 vs. SMEAR 1:

SMEAR 2 is an updated version of the SMEAR program which has been operational at SLAC since May, 1967. It provides more control over the OS utilities, more compile-linkedit options, and a more flexible and more mnemonic input format.

Some features of SMEAR 1 were not incorporated because of lack of use. In particular, the text editing commands have been dropped.

The major advantage of SMEAR 2, as of SMEAR 1, is the ease with which a user can maintain, update, and execute load modules using 231h disk storage. Beyond this are many minor advantages, some of which are not immediately apparent. For example, the ability to keep program source decks, program binary decks, and data decks stored on a disk, from which they can be listed and/or reproduced at will, may seem to be a small thing, but it has been surprisingly well appreciated by SMEAR users in the past.

The drastic reduction in card handling and minimization of linkedit time will easily pay for SMEAR operation overhead for the large program user. Small program users who but rarely make JCL errors (if, indeed, there are any such) will have to pay a small price in increased computer time for the convenience of using SMEAR. Methods to virtually eliminate even this small overhead, a matter of a few seconds per run for small programs, are discussed in the last section of this paper.

CRITICAL EVENTS

• Appropriate user disk pack is mounted on any convenient drive.

• A SMEAR scratch tape is mounted on tape unit CE2, a seldom used 7-track drive.
User drops in the hopper the execute SNEAR deck (supplied) followed by appropriate SNEAR control cards and source decks, if any, as required.

User stands well back.

SNEAR functions, creating JCL cards on the scratch tape for later input to the IBM utility and systems programs, and then flushes itself.

IBM utility and systems programs automatically operate off the tape onto the disk, under control of the SNEAR-created JCL cards.

User load module slithers into core from the disk.

User job executes normally (whatever that means).

DECK SET-UP AND OTHER CONSIDERATIONS

A normal SNEAR deck set-up is shown in block form in Figure 1 for an initial LOAD run. This run sets up separate source, object (binary), and load modules on the 2314 volume Yourdisk (which may be just a piece of a system disk). The GO command executes the newly created load module. Later runs to update and execute this load module would normally be set up as in Fig. 2. FULL SNEAR control card images are shown. Note that on an ADD run (figure 2) only new or modified subroutines are physically present in the deck; remaining unmodified subroutines are plucked from the old load module from disk storage before the new load module is created and stored back on the disk, where it replaces the old. Figure 3 is an actual (test) computer run. If the "FORTRAN.RES" option were removed from the LOAD card in Figure 3, the default compiler, FORTRAN.G, would be used. If the option "SEQUENCE" were not on this card, no source card sequencing would be performed.

Sequencing was made optional in SNEAR (as versus the old SNEAR) because some PDP users may want all 80 card columns for source code.

The SNEAR program performs a rather thorough examination of its input and rejects anything found indigestible. The error comments output by SNEAR are self-explanatory and, for that reason, have not been listed here. In general, SNEAR will accept any of the commands shown on page four, with options and data, in any sensible order. (Do I hear barely suppressed laughter?) Any given command may be repeated any number of times on any one SNEAR run. A SNEAR run may be logically reinitiated at any point by inserting in the deck the initial SNEAR command "SNEAR", the first command described on page 5.
FIGURE 1

SMEAR LOAD runs.

1. Trailing SMEAR deck, seq. #s 7 to 11
2. User data cards, if any
3. Special magnetic tape DB cards, if any
4. GO
5. Miscellaneous binary cards
6. LINK
7. Source cards
8. LOAD
9. SMEAR PROGRAM yourprogram ON yourdisk
10. Leading SMEAR deck, sequence #s 1 to 6

(This card is a standard system job card)
FIGURE 2

SMEAR ADD runs.

(trailing SMEAR deck, seq. #’s 1 to 1)

(user data cards, if any)

(Special magnetic tape DB cards, if any)

GO

(new or modified source subroutines or procedures)

ADD

SMEAR PROGRAM yourprogram ON yourdisk

(Leading SMEAR deck, sequence #’s 1 to 6)

(This card is a standard system job card)
This is the last card in the SMear deck

/*

START

/*

SMear Card follows all SMear control cards and data, if any.

LASTYTOC ON PUNCH
SCRATCH EVERYTHING
COPY TEST FROM ADA9O1 TO PUNOOL RENAME TEST.SAVE
PUNCH BINDING
PRINT SOURCE

THIS IS A DUMMY GARBAGE DATA CARD FOR THE TEST PROGRAM.

GO

END

STOP

2 FORMAT('1 TEST RUN INPUT CARD = ',20A1)

PRINT 2,ICARD

1 FORMAT('20A1')

READ 1,ICARD

DIMENSION ICARD(20)

C OPTIONS. IT READS IN AND PRINTS OUT OUR CARD.

C OUT FORTRAN COMPILE, LINKEDIT, AND EXECUTE

C MAIN PROGRAM. THIS IS A TEST PROGRAM DESIGNED SIMPLY TO CHECK

LOAD FORTRAN HZ SEQUENCE

SMear TEST OF ADA901

SYSTDD DD DATA

SMear control cards and data, if any, follow this card.

DCB=(DEFN=2,RECFM=EF,BSIZE=50,WTCH=C)

OF DD UNIT=(O82,DEFER),DISP=OL, VOLUME=S38-40666, LABEL=('ML'),

SYSPRINT DD SYSOUT=A

EXEC PROC=SMear

JUBLIB DD INCLUDE=SMear,DISP=OL,UNIT=O314, VOLUME=S38-ADA901

REXSMear JOB 'FX99ADA,5M,99P(ROCK BERMAM 3286)\',MSGLEVEL=1
SMEAR SYNTAX:

Let + mean exclusive or, and let ++ stand for concatenation. Then the structure of each SMEAR input card is:

\[ v ++ (bxb + bxbyy) ++ (bxb + bxbyy) ++ \ldots \]

where \( v \) is the SMEAR verb (which must start in column 1), \( b \) is one or more blanks, \( x \) is an item identifier (keyword), and \( y \) is a user input item. The parentheses do not appear on the card; they are used simply to delimit the scope of the ++ operator. As shown, some x's do not require a y, and blanks and used for delimiters. Because of the latter, there may be no blanks within an item identifier or item. The underscore symbol _ must be used in places where a blank would normally appear inside an item identifier. The item identifiers are reserved words only on the SMEAR control cards for which they are defined. The SMEAR verbs \( v \) are not reserved words. Item identifiers may appear in any columns, past the SMEAR verb itself, in any order. Items must follow their respective item identifiers. All SMEAR commands are completed in columns 1 to 80 of one physical card.

SMEAR COMMANDS:

Each command is shown with upper case symbols representing literals which must (or may) be present. Options are shown below each command. Note that each SMEAR verb must begin in card column one (1). After the verb, options, i.e. item identifiers and their associated items, if any, may appear in any card columns in any order. Comments may appear anywhere on a SMEAR control card after the first item, if any. A dollar sign ($) must start and end each comment; blanks may appear within a comment.
<table>
<thead>
<tr>
<th>Smear Commands</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMEAR</td>
<td>5</td>
</tr>
<tr>
<td>LOAD</td>
<td>5</td>
</tr>
<tr>
<td>LOAD-</td>
<td>7</td>
</tr>
<tr>
<td>ADD</td>
<td>7</td>
</tr>
<tr>
<td>ADD-</td>
<td>7</td>
</tr>
<tr>
<td>LINK</td>
<td>7</td>
</tr>
<tr>
<td>GO</td>
<td>8</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>9</td>
</tr>
<tr>
<td>PRINT</td>
<td>9</td>
</tr>
<tr>
<td>PUNCH</td>
<td>9</td>
</tr>
<tr>
<td>COPY</td>
<td>9</td>
</tr>
<tr>
<td>LISTYTOC</td>
<td>10</td>
</tr>
<tr>
<td>REPLACE</td>
<td>10</td>
</tr>
</tbody>
</table>
SMEAR Command:

SMEAR PROGRAM x ON y

Where:  x is an up to 8 character name we shall refer to as Yourprogram, and y is a 6 character name we shall call Yourdisk. The "ON y" is actually optional. If omitted, SMEAR will choose PDSPOOL and the user must then include a SCRATCH ALL card (described later), unless he has made arrangements with the systems people to reserve the necessary space. If LOAD- and ADD-, described later, are used, the user without a private disk pack will need only enough public disk space for his load module itself, since the source and object decks will not be saved.

This command (the SMEAR command) must be the first command for each SMEAR run. All other commands are optional. The item identifier or keyword PROGRAM may be omitted if the program name x is the first item on the card.

<table>
<thead>
<tr>
<th>Optional Item Identifiers</th>
<th>Item (if any)</th>
<th>Action</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARSX</td>
<td>-</td>
<td>SMEAR will print out its interpretation of each user input card suspected to be a, perhaps poorly thought out, SMEAR control card.</td>
<td>no such output print out</td>
</tr>
<tr>
<td>LST</td>
<td>-</td>
<td>All cards input to or created by SMEAR will be printed before being sacrificed to the ON 360 operating system.</td>
<td>no such output</td>
</tr>
</tbody>
</table>

LOAD Command:

LOAD

Action:  Yourprogram will be compiled (default FORTRAN G) and link edited (default R). The source cards will be stored on Yourdisk and named LOAD.SOURCE.Yourprogram. The object (binary) deck produced by the compiler will be stored on Yourdisk as LOAD.BINARIES.Yourprogram. The load module created by the linkage editor will be stored on Yourdisk simply named as Yourprogram. (The normal SMEAR user can simply ignore this information).

All of these data sets are first scratched at the beginning of the LOAD run. Standard compiler listings and link editor maps are output. Following these will be the listing of the volume table of contents of Yourdisk.
<table>
<thead>
<tr>
<th>Item Identifier</th>
<th>Item</th>
<th>Action</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN_G</td>
<td>*(see below)</td>
<td>The Fortran G compiler will be used</td>
<td>Fortran_G</td>
</tr>
<tr>
<td>FORTRAN_H</td>
<td>*(see below)</td>
<td>The Fortran H compiler will be used</td>
<td>&quot;</td>
</tr>
<tr>
<td>FORTRAN_NO</td>
<td>*(see below)</td>
<td>The Fortran H compiler optimisation level 2 will be used</td>
<td>&quot;</td>
</tr>
<tr>
<td>PLL</td>
<td>*(see below)</td>
<td>The PLL compiler will be used</td>
<td>&quot;</td>
</tr>
<tr>
<td>E</td>
<td>*(see below)</td>
<td>Link editor E will be used (44K)</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>*(see below)</td>
<td>Link editor F will be used (88K)</td>
<td>E</td>
</tr>
<tr>
<td>'options'</td>
<td></td>
<td>When following a compiler or link editor name, this is a complete list of options to be passed to that compiler or link editor. It must be enclosed in single quotes as shown.</td>
<td>Standard options are specified</td>
</tr>
<tr>
<td>NO_LINK</td>
<td></td>
<td>The link edit step will not be automatically set up. This would be used, e.g., if the user wanted to pass overlay instructions to the link editor via a following LINK command.</td>
<td>Standard link editing</td>
</tr>
<tr>
<td>026_T6_029</td>
<td></td>
<td>Source cards will be converted from 026 to 029 keypunch format.</td>
<td>No such conversion</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td></td>
<td>Source cards will be sequenced as they are stored on the disk in columns 73-80.</td>
<td>No sequencing</td>
</tr>
<tr>
<td>NO_COMPILE</td>
<td></td>
<td>Compiler will not be called. This option is generally used along with NO_LINK to store data decks on disk. Card images can be picked up from magnetic tape or other disk files by use of a suitable REPLACE command, described on page 10.</td>
<td>Specified or default compiler is called</td>
</tr>
</tbody>
</table>
LOAD- Command:

LOAD-

This command has the same action and options as LOAD except that only the load module itself will be saved on Yourdisk. (One or more blanks may appear between the verb and the minus sign.)

ADD Command:

ADD

Action: The source cards that follow will be compiled (default Fortran G) and link edited (default R) and standard listings output. Source cards will go to ADD.SOURCE.Yourprogram; compiler output will go to ADD.BINARIES.Yourprogram, both on Yourdisk. Both data sets are first scratched. (The normal SMEAR user can ignore these specific names). The compiler output is link edited into the yourprogram load module, which must exist from a previous LOAD run, thereby updating it to the new source cards. The volume tape of contents of Yourdisk will be listed at the end of the run.

The ADD command has the same item identifier options as the LOAD command. The main distinction between the LOAD and ADD command is that behind the ADD card one need put only the source subroutines (or procedures) which have been changed or newly created (i.e. coded) since the last ADD (or LOAD) run.

ADD- Command:

ADD-

Action: Same action and options as ADD except that the add.source and add.binary cards are not saved. (One or more blanks may appear between the verb and the minus sign.)

LINK Command:

LINK

Action: All binary cards which follow this card will be linked edited (default R) into the Yourprogram load module, if it exists. Linkage editor instructions following this card will be acted upon. In particular, linkage editor overlay cards may be included if the item identifier OVERLAY appears somewhere on this control card after the verb LINK.
Item Identifier Item Action Default
E 'options' The specified parameter list, E will get a Linkage editor
if any, will be passed to the standard parameter list linkage editor E
F 'options' The specified parameter list, if any, will be passed to the linkage editor F.
OVERLAY Tells the linkage editor that the Yourprogram load module is to be non overlay
Yourprogram load module is to be put into an overlay structure. linkededit

GO Command:
GO
Action: The Yourprogram load module will be executed. Any special DD cards required, if any (including in particular tape DD cards), may be placed anywhere after this card but before the first user data card, if any (make sure that "GO." is not part of the dname on these cards). Standard printer, card reader, card punch, and SYSUDUMP DD cards are automatically supplied. All user program data cards, if any, follow this card after all user DD cards, if any.

Item Identifier Item Action Default
PROGRAM x The up to 8 character program Yourprogram
name x is executed is executed
ON y This option is honored only if Joblib cards
PROGRAM x is also specified. for Yourprogram
It switches the joblib cards on Yourdisk to y are made up
from Yourdisk to y and used
SCRATCH, PRINT and PUNCH commands:

SCRATCH FILE x ON y
PRINT FILE x ON y
PUNCH FILE x ON y

Where: x is an up to 44 character data set name, counting indexing periods, if any, and y is a six character disk name. (The item identifier TREMOVE may be used for FILE for those who no longer have a choice.)

The "ON y" is optional with default Yourdisk. If x is the first or only item on the card, the item identifier FILE may be omitted.

For the scratch command, if x is the special symbol ALL or the special symbol EVERYTHING, then all data sets associated with Yourprogram will be deleted. This deletion will be done as a separate job, not job stop, to make certain it happens.

The print and punch commands work only on 80 character (i.e. card image) records. If x is the special symbol SOURCE, then the Yourprogram source cards (from file LOAD.SOURCE.Yourprogram) will be printed, or punched, or scratched. If x is the special symbol BINARY, then the Yourprogram binary cards (from file LOAD.BINARY.Yourprogram) will be punched, or scratched, or printed (!?).

COPY Command:

COPY FROM x TO y HERE u THERE v

Where: x and y are 6 character disk names, u and v are up to 44 character (counting indexing periods, if any) data set names.

Action: Data set u is copied from disk x to disk y where it is named v.

The "HERE v" is optional with default the "HERE u" name. If the "TO y" is omitted, "TO x" will be used. If both "FROM x" and "TO y" are omitted, "From Yourdisk" and "TO Yourdisk" will be used. The item identifier FILE may be used for HERE, e.g., "COPY FILE MY.DATA FROM PUB001 TO PUB002". If neither "FILE u" nor "HERE v" is specified, the default file Yourprogram will be copied from disk x to disk y where, if "THERE v" was given, it will be renamed v.

***Note: Since the above was written this command has been changed so that "FILE" must be used for "HERE" and "REMOVE" must be used for "THERE". The new form of the command is now:

COPY FILE u FROM x TO y REMOVE v

or

COPY u FROM x TO y REMOVE v

where order is important only in the second form and, of course, not all the options need be specified.
<table>
<thead>
<tr>
<th>Item Identifier</th>
<th>Item</th>
<th>Action</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td>z</td>
<td>Space in tracks of amount z is allocated to the data set on the T0 disk for the first and each additional (up to 16) extent.</td>
<td>20 tracks per extent (about 1,600 cards, very roughly)</td>
</tr>
<tr>
<td>HRRR_MEMBER</td>
<td>r</td>
<td>Only the member r will be picked up on the FROM disk</td>
<td>The full data set x is gathered</td>
</tr>
<tr>
<td>THERE_MEMBER</td>
<td>s</td>
<td>The HRRR_MEMBER will be renamed to s on the T0 disk</td>
<td>s = r</td>
</tr>
</tbody>
</table>

**LISTVTOC Command:**

**LISTVTOC**

Action: The volume table of contents on Yourdisk will be listed.

<table>
<thead>
<tr>
<th>Item Identifier</th>
<th>Item</th>
<th>Action</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>y</td>
<td>Volume table of contents on disk y will be listed</td>
<td>Yourdisk</td>
</tr>
</tbody>
</table>

**REPLACE Command:**

**REPLACE**

Action: Following this command will be one or more of the static JCL cards used by SUEAR, identified by ID-x somewhere in columns 73-80 (x is a 1 or 2 digit number). The ID number is shown for each such static JCL card on the output from any SUEAR run and can also be found in the SUEAR program source listing, below. Each such card encountered will immediately replace its corresponding card image in SUEAR, for the remainder of the current SUEAR run only. This command would be used for doing offbeat things such as storing compiler and linkeditor (and execute) print out on your disk where they may be later retrieved if the system fails, e.g., and so forth.

**SUEAR 2 Source Listing:**

The source listing for SUEAR 2 as of 1 March 1968 is reproduced below. Since this is my very first PII program, I hope you can find it in your heart to forgive any coding crudities you may encounter.
SYSS
NEW MASTER

DCL SYSPRINT FILE PRINT;
IBUG = 1;
LIST = 0;
/* BECAUSE OF GHASTLY PLI DEFAULT ATTRIBUTE ASSIGNMENT FOR INTEGER CONSTANTS*/
THE FOLLOWING KUDEGE WAS USED: /*
NO=0;1=1;2=2;3=3;4=4;5=5;6=6;7=7;8=8;9=9;10=10;15=15;77=77;80=80;
DCL IFITEM ENTRYCHAR(44) VARYING, BINARY FIXED(15,0));
SUBSTR(DOISEKAADB_2,70) = ' 10.19'; SUBSTR(DOEIEMAA_7) = ' 10.24';
/* CLEARING UP PROBLEM CAUSED BY NEEDING TWO SINGLE QUOTES TO STAND FOR
(ONLY) ONE IN A LITERAL CONSTANT */
*/
/* NOTE THAT MANY OF THE SEEMINGLY AWKWARD IF I THEN ELSE X BELOW
WERE USED BECAUSE IF ~ I THEN XI DOES NOT WORK AS ONE WOULD NAIVELY
ASSUME. IN FACT, IT IS THE LATTER STATEMENT THE BRANCH X WILL ALWAYS BE TAKEN
UNLESS I HAVE THE VALUE 2**N-1 FOR SOME N. */

FIRST_CARD : GET EDIT (ICD) (A(80));
ANOTHER_SMEAR_CARD: IBUG = 1; LIST = C; /* BOUNCE BACK TO HERE IF HIT NEW
SMEAR RUN CARD */
IF_LIST = 1
LOAD_RUN = 0; /* SETTING FLAG TO CONTROL THE INCLUDE SYSLMOD CARD OUTPUT */
IF_IFM = IFVERB('SMEAR '); PUT PAGE LIST (* SMEAR NOW HAS THE BALL*):
PUT SKIP(5) LIST(ICD):
IF I = 0 THEN BACFIRSTCARD; DO :
PUT LIST ('FIRST SMEAR CARD IS UNPALATABLE, BUTTERFINGERS,') PAGE :
PUT SKIP(3) DATA(IF_SMAR,IF_MYJOB,MY_DISK,IF_P1,IF_P2,IF_P3,IF_P4,IF_P5,IF/Observable/ICD); CALL CUMB_USER; STCP; END:
IF_LIST,LIST,IF_FOREACH = IFITEM('PARSER ','v6):
IF_P1 = IFITEM('P1 ',v6);
IF_P2 = IFITEM('P2 ',v6);
IF_P3 = IFITEM( P3, 'N6');
IF_P4 = IFITEM( P4, 'N6);
IF_PT = IFITEM( PT, 'N6');
IF IF_PT THEN: ELSE IF_PT = IFITEM( 'LIST', 'N6');
//PARSE CAUSES SMERF PARSE OF INPUT CONTROL CARDS TO BE LISTED */
//JT CAUSES DEBUG PRINTOUT IN PLUCK */
//JP2 CAUSES DEBUG PRINTOUT IN IFITEM */
#/P3 CAUSES DEBUG PRINT CUT IN READ_CARD */
#/P4 CAUSES JCL CARDS TO BE PRINTED AS THEY ARE OUTPUT ON OT */
#/PT OR LIST: CAUSES SHEAR OUTPUT TAPE CT TO BE PRINTED AT END OF RUN */
IF_DUMP = IFITEM( DUMP, 'N6'); /* DUMP CAUSES SPECIAL HEX DUMP OF DISK VTRC */
IF_IEBPDPTE = IFITEM( IEBUPDTPE, 'N6'); /* CAUSES NEW MASTER TO BE LISTED */
IF IFITEM( PROGRAM, 'G') THEN DO;
MYJOB = FINDI( PROGRAM, 'MYJOB'); GO TO NUMI; END;
DCL ME CHAR(44) VARYING; ME = FINDI(SHEAR, 'IGNORE');
IF ME = 'LIST' | ME = 'ALL' | ME = 'OPT' | ME = 'P3' | ME = 'P4' | ME = 'P1' | ME = 'DUMP' | ME = 'DB' | ME = 'ON' /* AND SO FORTH */
THEN DO; PUT PAGE LIST('MISSING PROGRAM NAME'); GO TO BADFIRSTCARD; END;
ELSE MYJOB = ME; /* ALLOWING USER TO OMIT ITEM IDENTIFIER PROGRAM. SEEMS LIKE A LOT OF WORK DOESNT IT? */
DUMI: I = 1;

MYDISK = FINDI( ON, 'MYDISK');
IF IF_MYDISK = O THEN MYDISK = 'PUBLIC';
DCL MYDISK CHAR(0) VARYING; IF_MEMBER = 0;
MEMBER = FINDI( MEMBER, 'MEMBER');
IF IF_MEMBER = O THEN MEMBER = 'GOGIRLGC';

IF LIST THEN PUT SKIP DATA(SHEAR, MYJOB, IF_MYDISK, IF_P1, IF_P2, IF_P3, IF_P4, IF_PT,
IF_LIST, MYJOB, MYDISK, IF_P1, IF_P2, IF_P3, IF_P4, IF_PT,
NEW_COMMAND = 'SHEAR';
OLDJOB = MYJOB;
IF SHEAR = 1: CALL START_JOB; IF SHEAR = O:

READ_SECOND_CARD: GET EDIT((LO1 A(IO1));

START: I = 1; /* MAIN PROCESSING CYCLE BEGINS HERE*/
IF LOAD.IF_LOAD_MINUS.IF_ADD.IF_ADD_MINUS.IF_FORTRAN.G I IF_FORTRAN.H, IF_PT1, IF_L...
/* DL LOAD OR ADD CHAR(5) VAR;
LAST_COMMAND = NOW_COMMAND; NOW_COMMAND = LOAD_OR_ADD;
IF LOAD, IF_LOAD_MINUS, IF_ADD, IF_ADD_MINUS, IF_FORTRAN_G, IF_FORTRAN_H, IF_PLI, IF_LINKEDITOR_F, IF_COMPILER_OPTIONS, IF_LINKEDITOR_OPTIONS, IF_NC_LINK, IF_P26_TO_P29, SEQUENCE 1;
IF LIST THEN PUT SKIP DATA (IF_FORTRAN_G, IF_LINKEDITOR_F, IF_COMPILER_OPTIONS);
IF_LIST & IF_LINKEDITOR_OPTIONS THEN PUT DATA (IF_LINKEDITOR_OPTIONS);
*/
SPEAR RUN */

CALL WRITE_DISK
IF IF_NO_COMPILE THEN: ELSE CALL COMPILE:

IF IF_NO_LINK = 0 THEN CALL LINKEDIT;
IF IF_LOAD_MINUS / IF_ADD_MINUS THEN CALL CLEAN_DISK/NO1;
CALL LIST_VTC31

GO TO START;
EAD LOAD OR_ADD_COMMAND;

1:IF_LINK = IFVERB('LINK'); IF I = I THEN LINK_COMMAND = DC:
PUT SKIP(3) LIST(IDC);
LAST_COMMAND = NEW_COMMAND; NEW_COMMAND = 'LINK';

CALL ITEM* E,*IF_LINK_EDITOR_F,IF_LINK_EDITOR_OPTIONS,IF_LINK_EDITOR_OPTIONS;
CALL ITEM* F,*IF_LINK_EDITOR_F,IF_LINK_EDITOR_OPTIONS,IF_LINK_EDITOR_OPTIONS;
IF IF_LINK_EDITOR_F THEN; ELSE IF_LINK_EDITOR_E = 1:
IF_OVERLAY = IFITEM(' OVERLAY ','5');

IF LIST THEN PUT SKIP DATA(ILINK,IF_LINK_EDITOR_E,IF_LINK_EDITOR_OPTIONS,
IF_LINK_EDITOR_F,IF_OVERLAY);
IF LIST E IF_LINK_EDITOR_OPTIONS THEN PUT SKIP DATA(ILINK,IF_LINK_EDITOR_OPTIONS);
CALL LINKEDIT;

GO TO START;
EAD LINK_COMMAND:

1:IF_GE = IFVERB('GO '); IF I = I THEN GO_COMMAND = DC:
PUT SKIP(5) LIST(IDC);
LAST_COMMAND = NEW_COMMAND; NEW_COMMAND = 'GO' ;
NEWJOB = FIND* PROGRAM *,IF_NEWJOB);
I = IFITEM* ON ,10); IF I E IF_NEWJOB THEN DISK = FIND* ON ,1,IF_DISK); ELSE DISK = MYDISK;
PUT SKIP(3) LIST(IDC);
IF IF_MEMBER = 0 THEN MEMBER = FIND* MEMBER *,IF_MEMBER);
IF IF_MEMBER = 0 THEN MEMBER = 'GOODNIGHT';

CALL EXECUTE;
IF LIST THEN PUT SKIP DATA(IF_GE,IF_NEWJOB);
IF LIST E IF_NEWJOB THEN PUT DATA(IF_NEWJOB);
IF LIST & IF_MEMBER THEN PUT SKIP DATA(MEMBER);
GO TO START;
END GC_COMMAND:

1. IF_COPY = *VERB*COPY ":
IF I ≠ 1 THEN COPY_COMMAND; DD;
LAST_COMMAND = NOW_COMMAND; NOW_COMMAND = *COPY*;
PUT SKIP(S) LIST(IDC);
IF_FROM_DISK, IF_TO_DISK, IF_HERE_NAME, IF_THERE_NAME, IF_HERE_MEMBER_NAME, IF_THERE_MEMBER_NAME, IF_SPACE = 0;

FROM_DISK = FIND( FROM *, IF_FROM_DISK);
IF_FROM_DISK = 0 THEN FROM_DISK = MYDISK;
TO_DISK = FIND( TO *, IF_TO_DISK);
IF_TO_DISK = 0 THEN TO_DISK = FROM_DISK;
IF_ITEM = *FILE *, *ITEM*, DSMAME *, *ITEM*, IF_HERE_NAME); THEN DD:
   HERE_NAME = FIND( FILE *, IF_HERE_NAME); IF_HERE_NAME THEN: ELSE
   HERE_NAME = FIND( DSMAME *, IF_HERE_NAME); GO TO DUMPS;
END:
ME = FIND(COPY *, IGNORE); IF ME ≠ 'FROM' | ME = 'TO' | ME = 'ALLOCATE' |
   ME = 'HERE_MEMBER' | ME = 'THERE_MEMBER' THEN
   HERE_NAME = MYJOB; ELSE
   HERE_NAME = FIND(COPY *, IF_HERE_NAME);

DUMPS: I = I+1
   HERE_NAME = FIND( RENAME *, IF_HERE_NAME); IF_HERE_NAME THEN: ELSE
   HERE_NAME = FIND( RENAME *, IF_HERE_NAME);
THERE_MEMBER_NAME = FIND( HERE_MEMBER *, IF_HERE_MEMBER_NAME);
THERE_MEMBER_NAME = FIND( HERE_MEMBER *, IF_HERE_MEMBER_NAME);
SPACE = FIND(ALLOCATE *, IF_SPACE);
IF_SPACE = 0 THEN SPACE = "2C";

CALL COPY;

IF LIST THEN PUT SKIP DATA(FROM_DISK, TO_DISK, HERE_NAME, THERE_NAME, SPACE);
IF LIST & IF_HERE_MEMBER_NAME THEN PUT DATA(HERE_MEMBER_NAME);
IF LIST & IF_THERE_MEMBER_NAME THEN PUT DATA(HERE_MEMBER_NAME);

GET EDIT(IDC) (AIBO); /* NEXT CARD MUST BE A SMAR CONTROL CARD */
GO TO START;
END COPY_COMMAND:

1. IF_SCRATCH = *VERB*SCRATCH ":
IF I = 1 THEN SCRATCH_COMMAND; DD;
LAST_COMMAND = NOW_COMMAND; NOW_COMMAND = *SCRATCH*;
IF_DATA, IF_DISK = 0;

CALL FIND_NAME_AND_DISK(SCRATCH *) ;

SYSIN
NEW MASTER
IF DSNAM= 'SOURCE' THEN DSNAM = 'LOAD.SOURCE.' || MYJOB;
IF DSNAM = 'BINARIES' THEN DSNAM = 'LOAD.BINARIES.' || MYJOB;

IF DSNAM = 'ALL' || DSNAM = 'EVERYTHING' THEN CRC: CALL START_JOB;
IF LOAD = 1: CALL CLEAN_DISK(); IF LOAD = 0: END;
ELSE CALL SCRATCH;

IF LIST THEN PUT SKIP DSNAM(DSNAM,DISK):
GET EDIT(ICS) (A(60)): /* NEXT CARD MUST BE A PDFER CONTROL CARD */
GC TO START;
END SCRATCH_COMMAND:

IF REPL = IFVERB('REPLACE '):
IF IF_REPL = 1 THEN REPLACE_COMMAND: CO;
LAST_COMMAND = NEW_COMMAND: NOW_COMMAND = 'REPLACE';
PUT SKIP(5) LISTEDC):

CALL REPLACE;

IF LIST THEN PUT SKIP DATA(IF_REPL):
GC TO START;
END REPLACE_COMMAND;

IF LISTVTCG = IFVERB('LISTVTG '):
IF IF_LISTVTCG = 1 THEN LISTVTG_COMMAND: CO;
LAST_COMMAND = NOW_Command: NOW_COMMAND = 'LISTVTG';
PUT SKIP(5) LISTEDC):

DISK = FIND(' ON ','IF_DISK');
IF IF_DISK = 0 THEN DISK = 'YDISK';
CALL LIST_VTG;
IF LIST THEN PUT SKIP DATA(IF_LISTVTG,IF_DISK,DISK):
GET EDIT(ICS) (A(60)); /* NEXT CARD MUST BE A PDFER CONTROL CARD */
GC TO START;
END LISTVTG_COMMAND:

IF PRINT = IFVERB('PRINT '):
IF PUNCH = IFVERB('PUNCH '):

IF IF_PRINT || IF_PUNCH THEN PRINT_EX_PUNCH_COMMAND: CO;
LAST_COMMAND = NEW_COMMAND;
IF IF_PRINT THEN NOW_COMMAND = 'PRINT';
IF IF_PUNCH THEN NOW_COMMAND = 'PLNCH';
PUT SKIP(5) LISTEDC):

CALL FIND_NAME_AND_DISK(NOW_COMMAND):
IF DSNAME = 'SOURCE' THEN DSNAME = 'ALL' THEN
   DSNAME = 'LOAD.SOURCE.' || MYJOB ;
IF DSNAME = 'BINARIES' THEN DSNAME = 'LOAD.BINARIES.' || MYJOB;

CALL PRINT_CR_PUNCH;

IF LIST THEN PUT SKIP DATA(IF_PUNCH,IF_PRINT,DSNAME,DISK);
GET EDITED) (A100) ; /* NEXT CARD MUST BE A SPEAR CONTROL CARD */
   GC TO START;
EAD PRINT_CR_PUNCH_COMMAND;

I = IF(VENB1('SHEAR ')); IF I = 1 THEN GO TO ANOTHER_SHEAR_CARD;

I,IF_SHEARED = IF(VENB1('SHEARED '));
IF I = 1 THEN QUIT: DGI PUT SKIP5) LIST('SHEARED');
LAST_COMMAND = NOW_COMMAND; NOW_COMMAND = 'SHEARED';
PUT PAGE LIST('IT** IS ALL OVER NEW**');
CALL WIND IT_UP: IF IF_PT THEN CALL PRINT_TAPE;
STOP: END QUIT;

GARBAGE_CARD: PUT PAGE LIST ('THE FOLLOWING IS A GARBAGE CARD. RELIVED.');
PUT LIST ('WE SHALL BE OUR VERY BEST TO IGNORE IT.' ) SKIP;
PUT SKIP4) LIST (ICD) ;
PUT SKIP4) LIST (' ') ;
GET EDITED) (A100) ; /* NEXT CARD MUST BE A SHEAR CONTROL CARD */
   GE TO START ;

/* THIS IS THE BLITTL END OF THE MAIN PROCESSING LCCP */
/* SPEAR PROCEDURES NOW FOLLOW IN ALPHABETICAL ORDER */

A: PROC ; /* A PUT AN AND SYMBOL INTO COLUMN 72 OF CARD IK */
   SUBSTRK(N72,N11) = '*';
END A;

CLEANDISK PROC(IALL) ; /* CALL CONTROLS WHETHER OR NOT THE LOAD MODULE ITSELF
   IS ALSO SCRATCHED */
   DISK = MYDISK;
   IF IF_LOAD I IF_LOAD_MINUS THEN DGI;
CONVETO_DP: PROC; /* CONVERTS 6 TC * TO 7 TC (< TO ) */

COMPIL: PROC; /* EXECUTES THE APPROPRIATE COMPILER USING LOAD.SOURCE, MYJOB, OR A DD.SOURCE, MYJOB FOR INPUT AND LLA.C,BINARIES, MYJOB, OR LLA.C, BINARIES, MYJOB, FOR OUTPUT */

IF IF_FORTRAN_C THEN IC = DICEFORT;
IF IF_FORTRAN_F THEN IC = DICEFAFD;
IF IF_FORTRAN_H2 THEN IC = ODIEKAAGD_2;
IF IF_FPL THEN IC = DDLFMAA;
IF IF_COMPILER_OPTIONS THEN OD;
DCL ISCRA CHAR(80); ISAVE = ISCRA; ICOD = IC;
DUMMY = PLUCKIN15,*,*END1;*
IF IF1 <= 15 THEN DUMMY = PLUCKIN15,*,*END2;*
ELSE SUBSTR(1K,*END1) = COMPILER_OPTIONS;
/* ABOVE NECESSARY AS PARM='...' ALREADY ON H2 & PLI CARD */

ICOD = ISCRA;

END;
END;
END;
END;
END;
END;
END;
END;
END;
END;
END;
END;
END;
IF IF_LOAD | IF_LOAD_MINUS THEN
   IK = SS15 | 'OSNAME=LOAD.BINARIES.' | MYJOB | '[(OBJECT);'
   IF IF_ADD | IF_ADD_MINUS THEN IK = SS15 | 'OSNAME=ADD.BINARIES.' | MYJOB | '[(
   BINARY)];'
   N = IK; CALL W;
   IK = 'SYSIN DD UNIT=2314,DISP=CLO,VOLUME=SER=* | MYDISK | *'; CALL A;
   N = IK; CALL W;
   IF IF_LOAD | IF_LOAD_MINUS THEN IK = SS15 | 'OSNAME=LOAD.SOURCE.' | MYJOB | '[(SOURCE)];'
   IF IF_ADD | IF_ADD_MINUS THEN IK = SS15 | 'OSNAME=ADD.SOURCE.' | MYJOB | '[(
   SOURCE)];'
   N = IK; CALL W;
   N = EEND; CALL W;
END COMPILE;

CONVERT: PROCIX(Y); DCL {X,Y} CHAR(44) VARYING;
            I = C;
            IF TID TO MAKE THIS MORE ELEGANT (AND FASTER) BY AUTOMATICALLY
            INCREASING SECOND PARAMETER OF IFITEM SO THAT ONLY THE REMAINING
            PORTION OF THE CARD, IF ANY, WOULD BE SCANNED ON EACH PASS. UNFORTUNATELY
            THIS CAUSED IFITEM TO START RETURNING PARDCP NUMBERS FOR ITS FUNCTIONAL VALUE,
            AS NEAR AS I COULD TELL. PRESUMABLY THIS HAS SOMETHING TO DO WITH THE
            FACT THAT IFITEM HAS VARYING LENGTH CHARACTER STRINGS AS PARAMETERS. */
            TOP: I = IFITEM(X,N); IF I THEN DO: SUBSTR(IDC,1,1) = Y; GO TO TOP; END;
            ELSE RETURN;
            END CONVERT;
            CALL CONVERT(*'*,**');
            CALL CONVERT(*',**');
            CALL CONVERT('**',*');
            CALL CONVERT(*',**');
END CONVERT_TO_O29;

COPY: PROC;
            /* GLPY MOVES DATA SET HERE_NAME FROM FROM_DISK TO TO_DISK WHERE
            IT IS CALLED THERE_NAME. IF IF_HERE_MEMBER_NAME THEN IT WILL PICK UP ONLY THE
            MEMBER HERE_MEMBER_NAME WHOSE ENTRY IN THE TID DISK WILL BE AN ENTRY MEMBER NAME CAM IS
            CREATING TO CONTAIN THE THERE NAMES. */
            N=DDTPEMOVE; CALL W;
            N=DDSYSPRT; CALL W;
            END MOVE_SYSUT; CALL W;
            N=DDSYSINS; CALL W;
            IK = '* COPY PDS=* | HERE_NAME | *';
            }
CALL A /* SLIDE SYMBOL & INTO COLUMN 72 OF CARD IK */
N=IKI CALL W:
# SS15 = ' ' /* TEMPORARY KLUDGE */
IK = SS15 ' FROM=2314 ' ' FROM_DISK ' ' TO=2314 ' ' TO_DISK ' ' ', '
CALL A /* SLIDE SYMBOL & INTO COLUMN 72 OF CARD IK */
N=IKI CALL W:
IF IF_HERE_MEMBER_NAME THEN THERE_NAME = THERE_NAME || ' ', '
IK = SS15 ' ' RENAME=' ' THERE_NAME:
IF IF_HERE_MEMBER_NAME THEN CALL A /* SLIDE SYMBOL & INTO COLUMN 72 OF CARD IK */
N=IKI CALL W:
IF IF_HERE_MEMBER_NAME THEN DO:
IF = SS15 ' ' SELECT=' ' THERE_MEMBER_NAME || ' ' THERE_MEMBER_NAME || ' ')
N=IKI CALL W: END;
N=IFKICZ CALL W:
SS15 = ' ' /* UNDOING TEMPORARY KLUDGE */
END COPY;

/* FOLLOWING ARE ALL (RATHER, ARE ALMOST ALL) THE STATIC DD CARDS USED BY
SPEAK. DD CARDS WHICH HAVE VARIABLE INFORMATION ARE MADE UP IN THE VARIOUS PRO
CEDURES CALLED FROM THE MAIN PROCESSING LOOP. */
DCL IK CHAR(80), SS15 CHAR(15) INIT('**');
DCL VOL SER CHAR(40) INIT('/*DISK DD UNIT=2314,Disp=CIO,Volume=SER*');
DCL DDEND Char(80) INIT('**');
DCL DDJOBLIB Char(80) INIT('joblib DD UNIT=2314,Disp=CIO');
N=IKI CALL W:
DCL DDJHPROM Char(80) INIT('EXEC PGM=JHPROMP';
DCL DDSYSPRINT Char(80) INIT('EXEC PGM=SYSPRINT';
DCL DDESGCLIB Char(80) INIT('EXEC PGM=DESGCLIB';
DCL DDSYSVCLIB Char(80) INIT('EXEC PGM=SYSVCLIB';
DCL DDSYSMACLIB Char(80) INIT('EXEC PGM=SYSMACLIB';
DCL DDSYSCMA Char(80) INIT('EXEC PGM=SYSCMA';
DCL DDSYSIN Char(80) INIT('EXEC PGM=SYSIN';
DCL DDSYSOUT Char(80) INIT('EXEC PGM=SYSOUT';
DCL DDSYSMOVE Char(80) INIT('EXEC PGM=SYSMOVE';
DCL DDSYSLOAD Char(80) INIT('EXEC PGM=SYSLOAD';
DCL DDSYSSPACE Char(80) INIT('EXEC PGM=SYSSPACE';
DCL DDSYSSPACE Char(80) INIT(0,7);
EXECUTE: PROC1 /* SETS UP A STEP TO EXECUTE MYJOB CN MYDISK IF NO PROGRAM NAME SPECIFIED. OTHERWISE A NEW N JOB IS CREATED TO EXECUTE NEWJOB ON 2314 VOLUME DISK */
IF 'IF NEWJOB = 0 THEN NEWJOB = MYJOB;
IK = '/* */ NEWJOB = ' NEWJOB = ' JOB X.X,MLEVEL=1';
N = IK; CALL W;
N = OFFAC; CALL W;
CALL WIND IT, U;
END DUMB_USER;

EXECUTE: PROC2 /* EXECUTE MYJOB CN MYDISK IF NO PROGRAM NAME SPECIFIED. OTHERWISE A NEW N JOB IS CREATED TO EXECUTE NEWJOB ON 2314 VOLUME DISK */
IF 'IF NEWJOB = 0 THEN NEWJOB = MYJOB;
IK = '/* */ NEWJOB = ' NEWJOB = ' JOB X.X,MLEVEL=1';
N = IK; CALL W;
N = OFFAC; CALL W;
CALL WIND IT, U;
END DUMB_USER;
N = DOSPRINT: CALL W;
N = LFPTOS: CALL W;
N = DFT06: CALL W;
N = DFT07: CALL W;
N = DESSLDMP: CALL W;

CALL SOURCE_CARDS:
N = CEND: CALL W;
ENC EXECUTE:

// AS THE SNAKE PROGRAM GREW, THERE CAME A TIME WHEN THE PLI COMPILER SUDDENLY LOST GRIP AND BEGAN FAILING. ALMOST ALL THE ERROR CODES IN THE CCA SERIES WERE OBTAINED AT ONE TIME OR ANOTHER, PLUS MANY MORE. RELUCTANTLY I DECIDED TO CUT THE PROGRAM IN TWO. THIS NECESSITATED MAKING UP THE EXTERNAL DECLARATIONS THAT FOLLOW AND Duplicating IN EACH HALF OF THE SEVERED PROGRAM. WHAT MORE CAN I SAY ? */

// POSTSCRIPT1: UNFORTUNATELY, ABOVE MANEUVER DID NOT HELP. */


// POSTSCRIPT3: FORCED TO REMOVE "EXT" FROM DECLARATIONS BELOW. */

// POSTSCRIPT4: THREW AWAY SOME 50 COD SUPERFLUOUS DCL CARDS FROM BELOW. MAY GOD GRANT THAT I DO NOT EVER REGRET THIS ACTION. */

DCL COMPILE_OPTIONS CHAR(42) VAR:
DCL DISK [char(5)] VAR:
DCL DSNAME CHAR(44) VAR:
DCL FCM_DISK CHAR(44) VAR:
DCL HERE_MEMBER_NAME CHAR(44) VAR:
DCL HERE_NAME CHAR(44) VAR:
DCL ICE CHAR(180) :
DCL LAST_COMMN Gonzalez CHAR(20) VAR;
DCL LINK_EDITOR_OPTIONS CHAR(42) VAR;
DCL MYDISK CHAR(6) VAR;
DCL OLDJOB CHAR(8) VAR;
DCL MYJOB CHAR(8) VAR;
DCL NEWJOB CHAR(8) VAR;
DCL NEW_COMMAND CHAR(20) VAR;
DCL SPACE CHAR(3) VAR;
DCL THERE_MEMBER_NAME CHAR(44) VAR;
DCL THERE_NAME CHAR(44) VAR;
DCL IF_DISK CHAR(6) VAR;

FIND: PROC(SYMBOL, IF_SYMBOL) CHAR(44) VARYING;
    /* RETURNS THAT ITEM, IF ANY, FOLLOWING THE ITEM IDENTIFIER SYMBOL ON THE
       INPUT CARD. */
   DCL (SYMBOL, PLUCK_NAME) CHAR(44) VARYING;
   IF SYMBOL = IFITEM(SYMBOL, 11);
       IF IF_SYMBOL THEN FIND_STRING_VALUE: DO;
       I = IF_SYMBOL + 1;
       DUMMY = NEXT_NON_BLANK();
       PLUCK_NAME = PLUCK(I, '*END); END FIND_STRING_VALUE;
       IF IF_SYMBOL = 0 THEN PLUCK_NAME = 'GARBAGE';
       RETURN (PLUCK_NAME);
   END FIND;
   DCL FIND_ENTRY RETURNS(CHAR(44) VARYING);
   DCL DUMMY CHAR(44) VARYING INIT(' ');

FIND_NAME_AND_DISK: PROC(VERB); DCL VERB CHAR(10) VARYING;
   IF IFITEM(* FILE, ' 11) IF IFITEM(* DSNAME, ' 11) THEN DO;
       DSNAME = FIND?(' FILE, *IF_DSNAME); IF IF_DSNAME THEN:
           ELSE DSNAME = FIND?(' DSNAME + 1); GO TO GUP2; END;
   IF FIND(VERB, IGNORE) = 'CA' THEN DO; PUT PAGE LIST(*NO DATA SET NAME*);
       GO TO GARBAGE_CARD; END;
   ELSE DSNAME = FIND(VERB, IF_DSNAME);
/* ABOVE ALLOWS USER TO OMIT KEYWORD FILE OR DSNAME */
   DUM2: I = 1;

SET_DISK: DISK = FIND(' CN ', IF_DISK);
   IF IF_DISK = 0 THEN DISK = MYDISK;
   END FIND_NAME_AND_DISK;

FIND_OPTIONS: PROC(M);
   OPTIONS = PLUCK(' , , *IF_DOC, END);
   IF SUBSTR OPTIONS, 1, 1 = ' ' THEN DO; PUT SKIP(5) LIST('MISSING RIGHT SING
       LE QUOTE MARK AFTER OPTIONS LIST, SUPER SPAS.');
       GO TO GARBAGE_CARD; END;
   END FIND_OPTIONS;
   DCL OPTIONS CHAR(50) VARYING;
IBM: PROC (IBM_NAME, IF_IBM_NAME, IBM_OPTIONS, IF_IBM_OPTIONS);
/* PICKS UP REQUESTED COMPILER OR LINK EDITOR NAME (IF PRESENT) AND COMPILER 
 OR LINK EDITOR OPTION LIST (IF PRESENT)
 DCL (IBM_NAME, IBM_OPTIONS) CHAR (44) VARYING;
 DCL APICSTRP B CHAR (3) VARYING INITI [" " ];
 APICSTRP = SUBSTR (APICSTRP, 2, 1);
 IF I = 0 THEN DO; IF SUBSTR (IBM_NAME, LENGTH (IBM_NAME), 1) = ", THEN
 SUBSTR (IBM_NAME, LENGTH (IBM_NAME), 1) = APICSTRP;
 /* USER CAN PUT OPTION LIST ',,' IMMEDIATELY AFTER COMPILER/EDITOR NAME */
 IF IF_IBM_NAME = IF_ITEN (IBM_NAME, 3); IF II THEN I = I; END;
 IF I THEN DO;
 IF NEXT (NUM, BLANK II) = APICSTRP THEN DO;
 IF_IBM_OPTIONS = I;
 IBM_OPTIONS = PLUCK (I, ' ', IEND);
 END; END;
 END IBM;

IFITEM: PROC (ITEM, BEGIN); /* CHECKS IF 
ITEM IS ON CARD FROM COLUMN BEGIN TO END OF CARD AND 
RETURNS IFITEM = LAST COLUMN OF ITEM IF YES, ELSE 0.*/
 DCL ITEM CHAR (44) VARYING; DCL ICARD CHAR (80) VARYING;
 IFITEM = BEGIN = ICARD; ITEM = ITEM;
 IF IF_IBN < 1 | (BEGIN > 80) THEN BEGIN = 1; 
 IF IBEGIN = 1 THEN ICARD = SUBSTR (ICARD, BEGIN); 
 ELSE ICARD = ICARD;
 I = INDEX (ICARD, ITEM); J = 0;
 IF IF/item = LENGTH (ITEM) - 1 + BEGIN = 1;
 IF IF_F2 THEN PUT SKIP DATA (ITEM, ITEMBEGIN, I, J);
 I = I + J; RETURN (ITEM); END IFITEM;

IFVERB: PROC (VERB);
/* CHECKS IF VERB IS A LEGITIMATE SPEARM
leo STARTING IN COLUMN CASE AND NC = 1 | 1 ;
ANYWHERE ON THE CARD */
 DCL VERB CHAR (10) VARYING;
 I = INDEX (ICARD, VERB);
 IF VERB = 'SPEARM' THEN RETURN (ITEM);
 IF I = 1 THEN; ELSE RETURNING) 0;
 IF ITEM = INDEX (ICARD, ' ' ); J = INDEX (ICARD, ' ' ); K = INDEX (ICARD, ' ' );
 DCL L BINARY; L = INDEX (ICARD, ' ' ); IF (L > 01) & (L < K) THEN K = 0;
 /* MUST PERMIT 'OPTION1,OPTION2; ... ' ON LOAD OR ADD CARD */
 IF II | J X THEN I = 0;
SYSLINK:PROC; /* IF A LEAD COMMAND, THEN LINK EDITOR PICKS UP LEAD BINARIES AND OUTPUTS TO A NEWLY CREATED MYJRP/INCLUDE SYSMOD CARD. IF AN ADD COMMAND, THE K ET PICKS UP THE ADD BINARIES AND THE OLD SYSMOD AND MAKES UP THE NEW. IF A LIN K COMMAND THEN IT PICKS UP BINARIES AND INSTRUCTIONS FROM THE CARD READER AND O UTPUTS TO SYSMOD */
IF IF_LINK_EDITOR_OPTIONS = 0 THEN LINK_EDITOR_OPTIONS = '-L(ET,MAP)';
IF IF_LINK_EDITOR_OPTIONS = 0 & IF_OVERLAY = 0 THEN LINK_EDITOR_OPTIONS = '-XREF,XCAL,LIST,DVLY)';
IF IF_LINK_EDITOR_E THEN
  IK = '# EXEC PGM=IEWLF440,PARM=* LINK_EDITOR_OPTIONS:
  IF IF_LINK_EDITOR_E THEN
    IK = '# EXEC PGM=IEWLF880,PARM=* LINK_EDITOR_OPTIONS:
N = IK; CALL W:
N = DDYSPRINT; CALL W:
N = DDLINKEDIT_SYSUTI:
CALL W:
N = ODDPRINT; CALL W:
N = DDLINKLIB; CALL W:
N = DDPLLIB; CALL W:
IK = 'SYSMOD DD UNIT=2314,DISP=OLD, N = DDLINKEDIT_OCB; CALL W; EN'D:
IK = 'SYSMOD DD UNIT=2314,DISP=OLD,UNIT=2314,VOLUME=SER=* MYDISK * DSNAME=* MYJOB ;
N = IK; CALL W:
IK = '#OBJECC DD DISP=OLD,UNIT=2314,VOLUME=SER=* MYDISK * DSNAME=* MYJOB ;
N = IK; CALL W:
DCL XTMP CHAR(44) VARYING; XTMP = 'NO-SHOW';
IF (NEW_COMMAND = 'LOAD') & (NEW_COMMAND = 'LINK') & (LAST_COMMAND = 'LOAD') THEN XTMP = 'LEAD.BINARIES.' MYJOB;
IF (NEW_COMMAND = 'ADD') & (NEW_COMMAND = 'LINK') & (LAST_COMMAND = 'ADD') THEN XTMP = 'ADD.BINARIES.' MYJOB;
IF XTMP = 'NO-SHOW' THEN XTMP = 'LOAD.BINARIES.' MYJOB ;
# LOAD OR ADD RUN MORE THAN ONE COMMAND BACK, SO WE TRY AND MAKE THE REST OF AN ESSENTIALLY WRITABLE SITUATION */
IK = SSS15 * DSNAME=* XTMP:
N = IK; CALL W:
N = DDYSLIN; CALL W:
IF IF_LINK > 0 & IF_OVERLAY = 0 THEN CALL SOURCE CARDS:
IF IF_LINK = 0 & IF_OVERLAY = 0 THEN DO:
  h = CDINCLUDE OBJECT; CALL W; EN'D:
IF (NEW_COMMAND = 'LOAD') & (LAST_COMMAND = 'LOAD') & (IF_OVERLAY = 0) THEN:
  ELSE DO:
    N = DDINCLUDE_SYSMOD; CALL W; EN'D:
IF (IF_LINK > 0) & (IF_OVERLAY > 0) THEN CALL SOURCE_CARDS;

/* REASON FOR THE ABOVE MANEUVER IS THAT THE LINKAGE EDITOR MUST GET THE INC.
LUCES CARDS BEFORE THE OVERLAY AND INSERT CARDS OR IT WILL PRODUCE INCORRECT
OUTPUT WITH NO WARNING TO USER. */

IF (IF_PJ) = 0 THEN DO: N=DEENTRY_MAIN CALL W: END:
IF (NGC_COMMAND = 'LOAD') | (LAST_COMMAND = 'LOAD') & (IF_OVERLAY = 0) THEN
  IK = * NAME * 1 MEMBER 1 4 1:
ELSE IK = * NAME * 11 MEMBER 11 4 1:
N = IK: CALL W:
N = CDARG: CALL W:
END LINKEDIT;

LIST_VTCC: PROC; /* LISTS VOLUME TABLE OF CONTENTS ON 2314 VOLUME DISK */
N = DDWTCPPXNT: CALL W:
I = 1:/VTOC IN DD UNIT=234,DISP=OLD,VOLUME=SER=* 11 DISK:
N = DDWTOCUT: CALL W:
N = DDEND: CALL W:
IF IF_CUMP = 0 THEN RETURN;

/* ON TO FULL VTOC PRINT OUT */
N = GDIEHLIST: CALL W:
N = GDSSYSPRINT: CALL W:
I = VCL,SER 11 DISK:
N=IK: CALL W:
N=DDSVSUSER: CALL W:
N=DDMACLIB: CALL W:
N=DDSSYEN: CALL W:
I = 1 LISTVTCC VCL=234=* 11 DISK:
N=IK: CALL W:
IF IF_CUMP THEN DUMP: DO:
I = 1 LISTVTCC DUMP,VCL=234=* 11 DISK:
N=IK: CALL W:
END DUMP:
N=DDEND: CALL W:
END LIST_VTCC;

NEXT_NON_BLANK: PROC(MARK) CHAR(1): /* RETURNS
FIRST NON BLANK CHARACTER FROM CARD COLUMN MARK ON.
MARK IS SET TO THE COLUMN WHERE THE FIRST NON
BLANK IS FOUND. IF GETS TO END OF CARD
SETS MARK TO 81 AND RETURNS A $ SIGN */
DCL I CHAR(11) DD J = MARK TC NBO:
I=SUBSTR(J, J ,1):

SYNIN
NEW MASTER
IF I = 1 THEN GO TO FOUND_IT;
END;
MARK=81: RETURN ("$"):
FOUND_IT: MARK = J: RETURN():
END NEXT_NON_BLANK:
DCL NEXT_NON_BLANK ENTRY RETURNS(CHAR(1)):

PLUCK: PRCC(ISTART,STOP_SYMBOL,IEND) CHAR(44) VARYING:
DCL STOP_SYMBOL CHAR(1);
DCL (STOP CHAR(I)); ISTART = ISTART; ISTOP = STOP_SYMBOL;
IF (ISTART < 1) OR (ISTART > 80) THEN ISTART = 1:
IEND = INDEX(SUBSTRING(I, ISTART), STOP_SYMBOL) + ISTART - 1:
IF IEND < IEND THEN PUT SKIP DATA(ISTART, ISTOP, IEND):
IF IF_EL_THEN PUT SKIP DATA(IINDEX, ISTART, IEND):
DCL TEMP CHAR(44) VARYING:
TEMP = SUBSTRING(I, ISTART, IEND - I:
DCL RETURN(SUBSTRING(I, ISTART, IEND - ISTART)):
/*THE ABOVE SHOULD RETURN WHAT bn CHARACTER STRING IS ON THE INPUT CARD FROM COLUMN ISTART TO THE FIRST COLUMN IN WHICH THE SPECIAL CHARACTER STOP_SYMBOL OCCURS (STOP_SYMBOL WILL USUALLY BE A BLANK) */
END:
DCL PLUCK ENTRY RETURNS(CHAR(44) VARYING):

PRINT_CB_PUNCH: PROC; /* PRINTS OR PUNCHES AN 80 CHARACTER BLOCKSIZE PDS */
N = DDIEPBPCH: CALL W:
N = DDIYSPRINT: CALL W:
N = DDISYCH: CALL W:
N = DDOMA: CALL W:
N = I=1: CALL W:
K = S/"SYSTAT DD UNIT=2314,DISP=OLD,DISK=SER=* DISK='*,':
CALL A1:
N = IK: CALL W:
K = SS15 || *OSNAME* || CSNAME:
N = IK: CALL W:
IF IF_print THEN DC N = DDIYSPRINT: CALL W: ENC:
ELSE IN: N = DDPUNCHUT2: CALL W: END:
N = DDISYCH: CALL W:
IF IF_PRINT THEN DO:
N = DDPUNCH: CALL W:
N = DDPYCH: CALL W:
END:
ELSE END:
N = DDPUNCH: CALL W:
END PRINT_CB_PUNCH:

PRINT_TAPE: PRCC /* PRINTS SMOAR OUTPUT TAPE GT */
/* AND SC FORTH */
END FCNLN_CKE:
IF IF_COMMAND = 1 THEN RETURN;
IF IF_COMMAND = 0 THEN GO TO GARBAGE_CARD;
END REPLACE:

SCRATCH: PROC: /*SCRATCH SIMPLY SCRATCHES DATA SET GSNAPF ON THE 2314 VOLUME DISK */
N=DDIEHPROMG: CALL W;
N=DDSYSPRINT: CALL W;
IK = VCLSeleccion || DISK;
N=IK (CALL W)
N=DDSVCIHI: CALL W;
N=DDMACLIB: CALL W;
N=DDSYSIN: CALL W;
IK = ' SCRATCH' || 'VCL=2314=' || DISK || 'Purge,' ;
CALL A: /* SLIDE SYMBOL 6 INTO COLUMN 72 OF CMD */
N=IK: CALL W;
IK = ' DSNAM=' || DSNAM;
N=IK: CALL W;
N=DDENC: CALL W;
END SCRATCH:

SOURCE_CARDS: PRCG : /* PICKS UP INPUT CARDS ONE AT A TIME AND IF NOT A SMEAR GO TOL CARD OUTPUTS ON GT. IF IF.GO AND THE LAST CARD BEGINS WITH '//' (IE IT IS A GC CARD) BUT EITHER THE NEXT CARD DOES NOT BEGIN WITH '//' (IE IT IS A USER DATA CARD) OR IT IS A SMEAR CONTROL CARD, THEN A DDSYSIN CARD IS OUTPUT ON GT. ALSO, A SYIN IS OUTPUT IF THE VERY FIRST CARD IS A USER DATA CARD */
LAST = NW = 0:
CALL READ_CARD;
IF IF.GO = 0 & (IF_COMMAND = 1) THEN PUT SKIP(1) LIST(1CD);
IF IF_LINK > 0 THEN PRINT_Overlay_CARDS: DO:
  IF (ITEM(1 ITEM(1 OVERLAY,1)) = 1) || (ITEM(1 INCLUDE "",1)) > 0
    THEN PUT SKIP LIST(1CD);
  END PRINT_Overlay_CARDS:
  IF IF.P3 THEN PUT SKIP DATA(IF_COMMAND, IF.GO);
  IF IF_COMMAND = I) & (IF.GO = 1) THEN G0; N=DDSYSIN: CALL W: RETURN: END;
IF IF_COMMAND = 1 THEN RETURN;
IF IF_COMMAND = 0 & (IF.GO = 1) THEN DG; N=DDSYSIN: CALL W; END;
N = IGC; CALL W;
NOW = IF_COMMAND;
READ_A_CARD: CALL READ_CARD; LAST = NW; NOG = IF_COMMAND;
IF IF.GO = 0 & IF_COMMAND = 1 THEN PUT SKIP LIST(1CD);
IF IF_LINK > 0 THEN PRINT_Overlay_CARDS: DG:
  IF (ITEM(1 ITEM(1 OVERLAY,1)) = 1) || (ITEM(1 INCLUDE "",1)) > 0
    THEN PUT SKIP LIST(1CD);
END PRINT_Overlay_CARDS: SO;

THEN PUT SKIP LIST(1CD); END PRINT_OVERLAY_CARDS2;
IF IF_P3 THEN PUT SKIP DATA(1;NOWN;IF_GO);
IF (LAST=2) & (OWN=0) & (IF_GO=1) THEN DO; N=DDSYSIN; CALL W; END;
IF (LAST=2) & (OWN=1) & (IF_GO=1) THEN DO; N=DDSYSIN; CALL W; END;
IF NOW = 1 THEN DO; N = ICD; CALL W; GO TO REAC_A_CARC; END;
ELSE RETURN;
END SELECT_CARDS;

START_JOB: PROC; /* CUTPLTS JOB CARDS TO SPEARM OUTPUT TAPE DT. OUTP
UTS OPERATOR RECOVERY PRECEDURE TO THE CONSOLE TYPEWRITER */;
1K = '/*' || MYJOB || ' JOB X, Y MSGLEVEL=1';
N = IKI CALL W;
IF B2JOP = MYJOB & IF_SPEARM = 0 THEN RETURN;
1K = '/* EXEC PGM=WTPARM,PARM=* IF IN TROUBLE, BELIEVE, CANCEL ' || MYJOB || '****;';
N=IKI CALL W;
1K = '/* EXEC PGM=WTPARM,PARM=* AND THEN START RCC, GDC** *';
N=IKI CALL W;
END START_JOB;

W: PROC; /* W WRITES OUT CARD IMAGE N ON OUTPUT TAPE DT */;
/* IF IF_P4 THEN THE CARD WILL BE PRINTED BEFORE IT IS WRITTEN ON DT */
IF IF_P4 THEN PUT SKIP LIST(1);
PUT FILE(CT) EDITIAL (A(BC));
END W;
DCL N CHAR(80);

WIND_IT_UP: PROC;
N=DDSTART_RDR; CALL W;
N=DDSTART; CALL W;
CLOSE FILE(CT);
ENW WIND_IT_UP;
DCL DDSTART CHAR(80) INIT('/** START*/);

WRITE_DISK: PKCG; /* PICKS UP SOURCE CARDS AND
SPEARS THEM ON MYDISK AS LOAD. SOURCE. MYJOB (IF
LOAD COMMAND), OR AS ADD. SOURCE. MYJOB (IF ADD
COMMAND), IF IF_SEQUENCE THE SOURCE CARDS ARE SEQUENCED
AS THEY GO ON, IF IF_D26_TO_C29 THEN CARDS
ARE CONVERTED FROM C26 TO C29 KEYPUNCH FORMAT */;
N=DDIEBUPDTE; CALL W;
IF IF_IEBUPDTE THEN N = DDSYSPRINT; ELSE N = DDUMMY; CALL W;
N=DDSYSWUMP; CALL W;
IK='//SYSUT2 UID=2314 VOLUME=SER='
" MYDISK " ': CALL A:
N=IK: CALL W:
N=ODECL: CALL W:
N=ODSPACE: CALL W:
N=OEXPIRATION_DATE: CALL W:
IF IF_LOAD_LOAD_MINUS THEN
IK=SS15 " 'OSNAME=LOAD.SOURCE." ' MYJOB:
IF IF_ADD_ADD_MINUS THEN
IK=SS15 " 'OSNAME=ADD.SOURCE." ' MYJOB:
N=IK: CALL W:
N=ODSYSIN: CALL W:
N=DDJUSTME: CALL W:
IF IF_SEQUENCE THEN DOIN=DDSEQUENCE_CARD: CALL W: END:
CALL SCURCE_CARDS:
IF C26_TO_C25 = C: /* IF ON WE NEED TURA IT OFF TO AVOID ANY UNWANTED CON-
VERSION LATER IN THE GAME */
N=ODENEUP: CALL W:
N=ODENEI: CALL W:
END WRITE_DISK:

END SPEAR:

 member name (justme) NOT FOUND IN NM DIRECTORY. SIGNED WITH 1TR.
 highest condition code was 00000000
 END OF JOB 1ERUPDATE.
SMEAR 2 improvements:

At the present time, no extensions to SMEAR 2 are immediately planned. When time becomes available, we will implement TAPE and FILE (for 2314 disk) cards with suitable modifiers and options to replace the DD cards now required on a GO run. The format could be pruned considerably from that of OS 360 JCL; and, simultaneously, the probability of error (enormously) reduced.

One very desirable SMEAR improvement is the replacement of the SMEAR output tape with a disk file. Unfortunately, this now requires OS system modifications of the same nature as implemented for the HASP job spooling program. When and if MVT gets here, this can (and will!) be done without effort simply by changing two cards in the execute SMEAR deck. Even more desirable is modifying SMEAR to call the OS utilities and systems programs directly (i.e. execute LINK macros), rather than via the present method of setting up JCL cards. Either of these improvements will virtually eliminate the overhead required for SMEAR execution, if someone will volunteer to do the work.

No doubt minor changes will be made to SMEAR 2 on a continuing basis as the needs of the ADA group evolve. We note here that minor extensions to SMEAR 2 can be made with truly remarkable ease. This is due in part to the open ended SMEAR 2 program design; however, this flexibility results primarily from the powerful P2L facilities for character string manipulation.