11. IBM UTILITIES

11.1 DFSORT

11.2 IEFBR14

11.3 IEBPTPCH

11.4 IEBGENER

11.5 IEBCOPY

11.6 IEHLIST

11.1 DFSORT

DFSORT is an IBM utility that can be used to sort a sequential data set or merge up to 32 sequential data sets. (VSAM data sets can also be hand- led; see the Programmers Guide). Additional features of DFSORT include reformatting and selecting or omitting specific records. Sequential data sets to be sorted or merged can be blocked or unblocked, and they can contain either fixed-length or variable-length records. A brief description of DFSORT follows.

The equivalent IBM utility is called DFSORT and has an essentially identical interface. DFSORT and SYNCSORT have a lot of similarities.

Module Name: SORT

Region: At least 65K; see the following PARM detail on 'CORE'.

PARM Values

1. CORE=nK

   The CORE parm is used to limit the amount of memory available to the sort within a given region. The default is all available memory within the region. Performance improves as the amount of main memory allocated to the utility increases.

2. CMP={CLC or CPD}

   The CMP parm identifies the compare instruction used when the fields being sorted contain numeric data. CLC specifies a logical compare is to be performed on the numeric data, which is used for non-packed data. CPD is used when comparing packed decimal data. If CPD is used and the data being compared is not packed decimal, an abend will occur (SOC7).

DD Statements

SYSIN: control statements (input data set)

SYSOUT: messages from the SORT utility, also returns a listing of the control statements received from SYSIN (output data set)

error message formats:

WERnnnA critical error (abnormal termination)
WERnnnI informational message
WERnnnD informational message regarding special options

SORTIN: the input data set to be sorted; the data set may be blocked or unblocked and may contain either fixed-length or variable-length records (record lengths must not exceed the track size of the intermediate storage device)

SORTOUT: the sorted output data set; the data set may be blocked (more

efficient) or unblocked and may contain either fixed-length or variable-length records; if no DCB parameter is specified, the data set is generated with the character- istics of the SORTIN data set (record lengths must not ex-ceed the track size of the intermediate storage device); variable-length records include the 4-byte Record Descrip-tor Word.

SORTWK01, SORTWK02, SORTWK03, . . ., SORTWK32:
scratchpad data sets to provide intermediate (virtual) storage to supplement main memory storage; the total number of tracks required for intermediate storage is approxi- mately 1.2 times the number of tracks required to hold the input data set, if it were stored on the same type of de- vice; for class assignments, one scratchpad data set of 5 tracks is sufficient.

1. **Control Statements**

The following control statement provides information concerning control fields and data set size. The control fields are the key fields on which the sorting is based.

[label] SORT FIELDS=(p1,l1,f1,o1[,p2,l2,f2,o2,. . .])
a. [,SKIPREC=r]
b. [,FILSZ=Ez]

- FIELDS: Up to 64 control fields may be specified. All of the control fields must be within the first 4092 bytes of the record (the sum of their lengths must not exceed 4092 bytes). Each control field has four characteristics:

p= the position of the control field within the record;
l= the length of the control field;
f= the control field data format, one of the following:

i. CH EBCDIC alphanumeric (unsigned, 1-4092 bytes)
ii. ZD numeric zoned decimal (signed, 1-256 bytes)
iii. PD numeric packed decimal (signed,1-256 bytes)
iv. FI numeric fixed-point (signed, 2-256 bytes)
v. BI numeric binary (unsigned, 1-4092 bits)
- = the collating sequence, one of the following:

vi. A ascending
vii. D descending

- SKIPREC=r
specifies that 'r' (number) records in the SORTIN data set (beginning with record number 1) are to be skipped before the sorting process begins.

- FILSZ=Ez
specifies the estimated number 'z' of records to be written to

the output data set. The more accurate the estimate, the better the performance of the sort.

1. The following control statement is required for variable-length record. records (and also for VSAM).

   [label] RECORD

2. The following control statement is used to limit the records input to the sort.

   [label] {INCLUDE or OMIT} COND=(comparison1[,{AND or OR}, comparison2,...])
   where comparison is

   p1,l1,f1,{EQ,NE,GT,GE,LT,LE},{p2,l2,f2 or self-defining term)

   and where p, l, and f are the position, the length, and the format respectively, as in the SORT control statement. The comparison fields for the INCLUDE or OMIT statement do not have to be control statements. INCLUDE and OMIT control statements cannot both be included in the same sort operation. The self-defining term may be used in place of a field from the input data set record. The self-defining term can be a signed or unsigned decimal number, a character string C'xxxxxxx', or a hexadecimal string X'hhhhhh'. There are restrictions on allowable comparisons. One basic rule to follow is to specify only data of the same type within one comparison.

SYSIN DD Statement Example

```
//SYSIN   DD *
        SORT   FIELDS=(2,4,CH,A,8,5,PD,D),
               SKIPREC=1,
               FILSZ=E400
        OMIT   COND=(1,1,BI,EQ,X'FF',OR,8,5,PD,LT,+1)
/*
```

Interpretation

Skip the first record. The remaining records are to be sorted in **A**scending order based on the **CH**aracter string in the 4-byte field starting in position 2. In case of a tie in this field, arrange these records in **D**escending order based on the 5-byte, **P**acked **D**ecimal field starting in position 8. However, any record containing either X'FF' in the first byte or having a **P**acked **D**ecimal value of less than 1 in the 5-byte field starting in position 8 is not to be placed into the output data set. The output file will contain an **E**stimated 400 records.

DFSORT Example 1

```
//KC0nnnnA JOB ,'DFSORT EXAMPLE 1',MSGCLASS=H
//*
//*
```

```
//*
//SORTIT   EXEC PGM=SORT,PARM='CMP=CLC'
//*
//*
//*********************************************************************
//*                                                                   *
//*  'SORTIT' INVOKES DFSORT TO SORT AN INSTREAM DATA SET             *
//*  WITH THE FOLLOWING FORMAT:                                       *
//*                                                                   *
//*  LONGEST RIVERS OF THE WORLD                                      *
//*  (SOURCE OF INFORMATION:  THE HAMMOND WORLD ATLAS)                *
//*                                                                   *
//*  COLUMNS    CONTENTS                                              *
//*                                                                   *
//*  01-29      RIVER                                                 *
//*  30-42      LOCATION                                              *
//*  43-46      LENGTH IN MILES                                       *
//*  47-50      LENGTH IN KILOMETERS                                  *
//*                                                                   *
//*  DD NAMES:                                                        *
//*                                                                   *
//*  SORTIN    THE INPUT DATA SET TO BE SORTED                        *
//*  SORTOUT   THE SORTED OUTPUT DATA SET                             *
//*  SYSIN     CONTROL STATEMENTS (INPUT)                             *
//*  SYSOUT    MESSAGES FROM THE UTILITY (OUTPUT)                     *
//*                                                                   *
//*********************************************************************
//*
//*
//SORTOUT  DD SYSOUT=*
//*
//SYSOUT   DD SYSOUT=*
//*
//SYSIN    DD *
  SORT FIELDS=(1,29,CH,A),
       FILSZ=E20
/*
//*
//*********************************************************************
//*                                                                   *
//*  'SORTWK01' IS A TEMPORARY WORK DATA SET REQUIRED BY THE          *
//*   UTILITY.                                                        *
//*                                                                   *
//*********************************************************************
//*
//SORTWK01 DD DSN=&&SORTWK01,
//
//            DISP=(NEW,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//*
//SORTIN  DD *
```

```
NILE                          AFRICA       41456671
AMAZON                        SOUTH AMERICA39156300
CHANG JIANG (YANGTZE)         CHINA        39006276
MISSISSIPPI-MISSOURI-RED ROCKU.S.A.        37416019
OB'IRTYSH-BLACK IRTYSH        U.S.S.R.     33625411
YENISEY-ANGARA                U.S.S.R.     31004989
HUANG HE (YELLOW)             CHINA        28774630
AMUR-SHILKA-ONON              ASIA         27444416
LENA                          U.S.S.R.     27344400
CONGO (ZAIRE)                 AFRICA       27184374
MACKENZIE-PEACE-FINLAY        CANADA       26354241
MEKONG                        ASIA         26104200
MISSOURI-RED ROCK             U.S.A.       25644125
NIGER                         AFRICA       25484101
PARANA-LA PLATA               SOUTH AMERICA24503943
MISSISSIPPI                   U.S.A.       23483778
MURRAY-DARLING                AUSTRALIA    23103718
VOLGA                         U.S.S.R.     21943531
MADEIRA                       SOUTH AMERICA20133240
/*
//
```

DFSORT Example 2

```
//KC0nnnnA JOB ,'DFSORT EXAMPLE 2',MSGCLASS=H
//*
//*
//*
//SORTIT   EXEC PGM=SORT,PARM='CMP=CPD'
//*
//*
//*****************************************************************
//*                                                               *
//*  'SORTIT' INVOKES DFSORT TO SORT AN INSTREAM DATA SET         *
//*  WITH THE FOLLOWING FORMAT:                                   *
//*                                                               *
//*  LONGEST RIVERS OF THE WORLD                                  *
//*  (SOURCE OF INFORMATION:  THE HAMMOND WORLD ATLAS)            *
//*                                                               *
//*  COLUMNS    CONTENTS                                          *
//*                                                               *
//*  01-29      RIVER                                             *
//*  30-42      LOCATION                                          *
//*  43-46      LENGTH IN MILES        (PACKED DECIMAL)           *
//*  47-50      LENGTH IN KILOMETERS   (PACKED DECIMAL)           *
//*                                                               *
//*  DD NAMES:                                                    *
//*                                                               *
//*  SORTIN    THE INPUT DATA SET TO BE SORTED                    *
//*  SORTOUT   THE SORTED OUTPUT DATA SET                         *
//*  SYSIN     CONTROL STATEMENTS (INPUT)                         *
```

```
//*  SYSOUT    MESSAGES FROM THE UTILITY (OUTPUT)              *
//*                                                            *
//************************************************************
//*
//*
//SORTIN   DD DSN=KC0nnnn.RIVER.DATA,
//            DISP=(OLD,KEEP,KEEP)
//*
//SORTOUT  DD DSN=&&RIVERS,
//            SPACE=(TRK,(1,1)),
//            DISP=(NEW,PASS,DELETE)
//*
//SYSOUT   DD SYSOUT=*
//*
//SYSIN    DD *
  SORT FIELDS=(43,3,PD,A),
       FILSZ=E20
/*
//*
//************************************************************
//*                                                          *
//*  'SORTWK01' IS A TEMPORARY WORK DATA SET REQUIRED BY THE  *
//*  UTILITY.                                                *
//*                                                          *
//************************************************************
//*
//SORTWK01 DD DSN=&&SORTWK01,
//
//            DISP=(NEW,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*
//************************************************************
//*                                                          *
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT THE SORTED FILE.    *
//*                                                          *
//************************************************************
//*
//*
//SYSUT1   DD DSN=&&RIVERS,
//            DISP=(OLD,PASS,DELETE)
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  PRINT    TYPORG=PS,     PHYSICAL SEQUENTIAL FILE ORGANIZATION
```

```
          CNTRL=2,       DOUBLE SPACING
          MAXFLDS=4,     FOUR FIELD OPERANDS ON RECORD OPERATION
  TITLE   ITEM=('WORLD''S  20  LONGEST RIVERS',30)
  TITLE   ITEM=('RIVER',11),
          ITEM=('LOCATION',39),
          ITEM=('MILES',53),
          ITEM=('KILOMETERS',62)
  RECORD  FIELD=(29,1,,4),
          FIELD=(13,30,,37),
          FIELD=(3,43,PZ,53),
          FIELD=(3,46,PZ,64)
/*
//
```

DFSORT Example 3

The following is another example of a use of the IBM DFSORT utility:

```
//JSTEP0n  EXEC PGM=SORT,PARM='CMP=CLC'
//*
//SORTIN   DD DSN=KC0nnnn.CSCI465.xxxx(yyyy),DISP=SHR
//SORTOUT  DD DSN=&&SORTED,SPACE=(TRK,(1,5)),DISP=(NEW,PASS)
//*
//SYSOUT   DD SYSOUT=*
//*
//SYSIN    DD *
  SORT FIELDS=(1,2,ZD,A)
/*
//
```

The SORTIN data set is being sorted in ascending order on a single two-byte zoned decimal field.

The SYSIN above is an instream control card telling DFSORT which bytes to sort on. The first positional parameter gives the byte in each record where the sort field begins, the second gives the length in bytes of the sort field, the third gives the type of data being sorted (ZD=zoned decimal) and the fourth gives the direction of the sort where A=ascending collating sequence and D=descending collating sequence.

By the way, what does the CLC in the PARM='CMP=CLC' mean?

A secondary sort field can be specified by adding four more positional parameters specifying the beginning byte, field length in bytes, data type and direction of the sort.

A tertiary or quaternary sort could be added too!

11.2 IEFBR14

The IEFBR14 utility is an Assembly language program consisting of only one instruction:

```
      BR    14
```

This instruction is an exit statement. The purpose of executing the IEFBR14 program is to con the operating system into processing the DD statements included in the job step which executes the IEFBR14 program. For non-VSAM data sets, IEFBR14 can be executed so that the operating system will process DD statements to create, delete, catalog, and uncatalog data sets.

The action taken by the operating system is determined by the disposition (DISP) parameter on the DD statements included in the job step. If the current disposition is (NEW,KEEP), the operating system allocates space for the data set and places an appropriate entry in the VTOC for the designated volume. A data set that existed prior to the execution of the job step can be deleted by specifying a disposition of (OLD,DELETE) for normal termination. The operating system deletes the corresponding VTOC entry and frees the space previously occupied by the deleted data set. A data set can be cataloged or uncataloged by specifying the appropriate disposition.

Thus, a data set can be created by including a DD statement with a current disposition of new in the IEFBR14 job step. After execution of the IEFBR14 step, the data set will exist since space will have been allocated for it by the operating system (according to the SPACE parameter). Subsequent DD statements for the data set would then specify a current disposition of either OLD, MOD, or SHR.

A data set can be deleted during execution of the IEFBR14 job step by including a DD statement with a current disposition of either OLD, MOD, or SHR and a normal termination disposition of DELETE. After execution of the job step, the data set will no longer exist.

Multiple data sets can be created, cataloged, uncataloged, and/or deleted by the operating system in one job step with the execution of IEFBR14. The operating system will not process the DD statements unless a program is executed in the job step. This is why IEFBR14 is executed even though it does nothing but return control to the operating system. The source code for IEFBR14 looks like this:

```
IEFBR14  CSECT
         BR    14
```

IEFBR14 Example

```
//KC0nnnnA JOB ,'IEFBR14 EXAMPLE',MSGCLASS=H
//*
//BR14STEP EXEC PGM=IEFBR14
//*
//DATASET1 DD DSN=DSN1,DISP=(OLD,DELETE,DELETE)
//*
//DATASET3 DD DSN=DSN3,DISP=(OLD,CATLG,KEEP)
//*
//DATASET4 DD DSN=DSN4,SPACE=(TRK,(1,1,2),DISP=(NEW,KEEP,DELETE)
//
```

Notes

`DATASET1` already exists and will be deleted.
`DATASET3` already exists and will be kept.
`DATASET4` will be created (and catalogued by SMS).

11.3 IEBPTPCH

IEBPTPCH is a utility which takes records from an input data set, SYSUT1, and writes all or some of them, as they are or with editing, to an output data set, SYSUT2, which may be the printer, a card punch, a sequential data set, or a member of a PDS.

We can arrange to skip header records, print all records, or print at most a specified number of records. The editing can include omitting fields, rearranging fields, or converting numeric fields from one format to another. The output can be printed with a page heading and column headings, and it can be printed with single, double or triple spacing.

The most common use of IEBPTPCH is to print a copy of a data set.

These notes do not cover any card punch possibilities, and some other options are also not covered. IEBPTPCH has much in common with IEBGENER.

JCL for IEBPTPCH

```
//JSTEP01  EXEC PGM=IEBPTPCH
//SYSUT1   DD DSN=...
//SYSUT2   DD DSN=...
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   various control statements if needed
/*
//
```

Required DD Statements

`SYSUT1` – input file. This may be in-stream data (but why?), a sequential data set, a member of a PDS, or concatenated data sets having similar `LRECL`, etc.

`SYSUT2` – output file. This is normally the printer, but need not be; we can print to a data set and print the data set later. `RECFM=FBA` or `RECFM=FBM`. `LRECL` may be between 2 and 145; the default is 121. The length includes the carriage-control character.

`SYSPRINT` – printed output generated by `IEBPTPCH` itself such as messages, acknowledgements, etc. `LRECL=121` and `RECFM=FBA`.

`SYSIN` – control statements. `LRECL=80` and `RECFM=FB`. This is normally in-stream data but need not be such.

Control Statements

`PRINT`

This indicates that the output is to be printed.

```
      PRINT     [PREFORM=c]
                [,TYPORG=xx]
```

```
[,TOTCONV=yy]
[,CNTRL=k]
[,STRTAFT=m1]
[,STOPAFT=m2]
[,SKIP=p]
[,MAXNAME=n1]
[,MAXFLDS=n2]
[,MAXGPS=n3]
[,MAXLITS=n4]
[,INITPG=q]
[,MAXLINE=n5]
```

`PREFORM=A` or `M` where `A` indicates that the first character of each record contains an ASA carriage control character and `M` indicates that the first character of each record contains a machine-code control character. That is, the file has already been formatted, perhaps by using `IEBPTPCH`. If the input record is longer than the output line, the record may span several lines, all single-spaced after the first. The default is neither. Note: If `PREFORM` is used, all other parameters except `TYPORG` are ignored.

`TYPORG=xx` where `xx` is either `PO` or `PS`. Here `PO` indicates that `SYSUT1` is partitioned (a member of a PDS), and `PS` indicates that `SYSUT1` is a sequential data set. Default: `PS`.

`TOTCONV=yy=either XE or PZ.` This controls the conversion of data and can be overriden by later RECORD statements. XE indicates that data should be printed in 2-character-per-byte format, and PZ indicates that packed decimal numbers should beconverted to zoned decimal numbers. Default: neither.

`CNTRL=k.` This specifies line spacing for the output. Here k=1 indicates single spacing, k=2 indicates double spacing, and k=3 indicates triple-spacing.

`STRTAFT=ml` indicates that `ml` records (`ml <= 32767`) should be skipped before printing begins. Default: skip nothing.

`STOPAFT=m2` indicates that m2 records (ml <= 32767) should be printed. Default: print everything.

`SKIP=p` indicates that every pth record is to be printed. Default: print every record.

`MAXNAME=n1 >=` the number of member names appearing on subsequent MEMBER statements. This is required if there are any MEMBER statements at all. Here n1 <= 32767.

`MAXFLDS=n2 >=` the number of FIELD parameters appearing on subsequent `RECORD statements. This is required if there are any FIELD statements at all. Here n2 <= 32767.`

`MAXGPS=n3 >= the number of IDENT parameters appearing on subsequent RECORD statements. This is required if there are any IDENT statements at all. Here n3 <= 32767.`

`MAXLITS=n4 >= the total number of characters appearing in the IDENT literals of any subsequent RECORD statements. This is required if there are any literals at all. Here n4 <= 32767.`

`INITPG=q` where q is integer between 1 and 9999. This is the initial page numbers. Pages are numbered sequentially. The default is 1.

MAXLINE=n5=the maximum number of lines to be printed on a page. Default: 60.

TITLE (optional parameter)

This is used to print a title and/of column headings as the first two lines of each page of output.

```
      TITLE  ITEM=('title'[,m])[,(ITEM=...)]
```

Here the string 'title', which may be up to 40 characters long, is printed at the top of each page, starting in column m. The default value of m is 1.

We may have multiple ITEMs on a TITLE statement, and we may have up to 2 TITLE statements.

MEMBER

This statement identifies members of a PDS which we want to print.

```
      MEMBER NAME=membername
    NAME=membername gives the actual name of the PDS member.
```

We may have any number of MEMBER statements; if we have none, and we are dealing with a PDS, we will print all members.

RECORD

Each MEMBER statement may be followed by RECORD statements until the next MEMBER statement is encountered. A RECORD statement is needed if the input is partitioned, or if editing is to be performed. We may have RECORD statements without a MEMBER statement if we are editing a sequential data set.

A RECORD statement defines a "record group", that is, a group of consecutive records to be processed identically. The last record in the group is identified by matching some part of it with a literal. If there is no RECORD statement, the entire data set is processed as one group.

```
      RECORD  [IDENT=(m1,'name',m2)]
              [FIELD=(n1,[n2],[conversion],[n3])],
              [FIELD.....]
```

IDENT is used to identify the last record in a group. It matches a field in the record, of length m1 bytes, starting at input location m2, that is, column m2, with the value specified in the literal 'name'. Here m1 <= 8.

FIELD is used for editing. There may be numerous FIELD statements, as each and every byte in the output record may be specified. The effect is as follows: take a field from the input record of length n1 bytes, starting at input location n2 (that is, column n2), optionally convert it as specified, and write it in the output record starting at output location n3, that is, starting in column n3. Default: n2=1 and n3=1. Notice that n1 is required.

The available conversions include:

(none)	no conversion
PZ	convert packed decimal numbers to unpacked decimal numbers
XE	convert alphanumeric data to hexadecimal data (which doubles the length of the field)

There are other conversions available as well. The default conversion is no conversion at all. In this case, code two consecutive commas. Notice that we may change the width of a field when we convert it.

Printing a Member of a PDS

There are two ways to do this:

1. Specify the name of the PDS in the SYSUT1 DD DSN parameter, use TYPORG=PO, and list the name of the desired member in a MEMBER NAME line; or

2. Specify the member name in parentheses after the PDS name in the SYSUT1 DD DSN parameter and use TYPORG=PS.

Either of these should work. In either case, a RECORD FIELD line is needed to provide the record length (at least).

Special Cases

(a) Print a PDS directory:

The directory of a PDS is considered a sequential data set for this purpose. (Often much of a PDS directory will not be printable.) To print it:

1. Specify TYPORG=PS.
2. Specify RECFM=U, BLKSIZE=256, and LRECL=256 on the SYSUT1 DD statement.

(b) Default format:

To print records of unknown length, or too long to fit on one line, specify PRINT and do not use a FIELD statement. The default format is:

1. Each logical record begins on a new line in the output, delimited by '*'.
2. Each block is delimited by '**'
3. Each printed line contains groups of 8 characters separated by 2 blanks. There may be 12 up to groups per line, containing 96 characters, 22 inserted blanks and '*' or '**'; if the logical record is longer, the printing continues on the next line.
4. We have 60 lines per page.
5. Non-printable characters are printed as blanks.

Examples

Two examples of IEBPTPCH are on the following pages.

Example 1 prints a data set using a page heading, column headings, double spacing, and some editing including conversion.

Example 2 prints a data set in the default format using two lines of page heading and single spacing.

Two more such examples can be found in the section on IEBGENER.

IEBPTPCH Example 1

```
//KC0nnnnA JOB ,'IEBPTPCH EXAMPLE 1',MSGCLASS=H
//*
//*
//*****************************************************************
//*  THIS JOB PRINTS A FORMATTED LISTING OF A DATA SET FILE     *
//*  USING THE IBM UTILITY 'IEBPTPCH'.                          *
//*****************************************************************
//*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*****************************************************************
//*                                                               *
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT A DATA SET FILE WITH   *
//*  THE FOLLOWING FORMAT:                                        *
//*                                                               *
//*  PRINCIPAL MOUNTAINS OF THE WORLD                             *
//*  (SOURCE OF INFORMATION:  THE HAMMOND WORLD ATLAS             *
//*                                                               *
//*  COLUMNS    CONTENTS                                          *
//*  01-18      MOUNTAIN                                          *
//*  19-32      LOCATION                                          *
//*  33-35      HEIGHT IN FEET    (PACKED DECIMAL)                *
//*  36-38      HEIGHT IN METERS  (PACKED DECIMAL)                *
//*                                                               *
//*  DD NAMES:                                                    *
//*  SYSUT1     THE INPUT DATA SET                                *
//*  SYSUT2     THE OUTPUT DATA SET                               *
//*  SYSIN      CONTROL STATEMENTS (INPUT)                        *
//*  SYSPRINT   MESSAGES FROM THE UTILITY (OUTPUT)                *
//*                                                               *
//*****************************************************************
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
```

```
//SYSIN    DD *
  PRINT    TYPORG=PS,     PHYSICAL SEQUENTIAL FILE ORGANIZATION
           CNTRL=2,       DOUBLE SPACING
           MAXFLDS=4,     FOUR FIELD OPERANDS ON RECORD OPERATION
  TITLE    ITEM=('WORLD''S  20  TALLEST  MOUNTAINS',20)
  TITLE    ITEM=('MOUNTAIN',7),
           ITEM=('LOCATION',27),
           ITEM=('FEET',44),
           ITEM=('METERS',52)
  RECORD   FIELD=(18,1,,4),
           FIELD=(14,19,,26),
           FIELD=(3,33,PZ,43),
           FIELD=(3,36,PZ,53)
/*
//*
//SYSUT1  DD DSN=KC03CF7.MOUNTAIN.DATA,UNIT=DISK,
//           DISP=(OLD,KEEP,KEEP)
//*
//
```

IEBPTPCH Example 2

```
//KC0nnnnA JOB ,'IEBPTPCH EXAMPLE 2',MSGCLASS=H
//*
//*
//*****************************************************************
//*                                                               *
//*  THIS JOB PRINTS A UNFORMATTED LISTING OF A DATA SET FILE     *
//*  USING THE IBM UTILITY 'IEBPTPCH'.                            *
//*                                                               *
//*****************************************************************
//*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*
//*****************************************************************
//*                                                               *
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT A DATA SET FILE WITH     *
//*  THE FOLLOWING FORMAT:                                        *
//*                                                               *
//*  PRINCIPAL MOUNTAINS OF THE WORLD                             *
//*  (SOURCE OF INFORMATION: THE HAMMOND WORLD ATLAS)             *
//*                                                               *
//*  COLUMNS    CONTENTS                                          *
//*                                                               *
//*  01-18      MOUNTAIN                                          *
//*  19-32      LOCATION                                          *
//*  33-35      HEIGHT IN FEET    (PACKED DECIMAL)                *
//*  36-38      HEIGHT IN METERS  (PACKED DECIMAL)                *
//*                                                               *
```

```
//*  DD NAMES:                                                    *
//*                                                               *
//*  SYSUT1    THE INPUT DATA SET                                 *
//*  SYSUT2    THE OUTPUT DATA SET                                *
//*  SYSIN     CONTROL STATEMENTS (INPUT)                         *
//*  SYSPRINT MESSAGES FROM THE UTILITY (OUTPUT)                  *
//*                                                               *
//***************************************************************
//*
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  PRINT    TYPORG=PS,     PHYSICAL SEQUENTIAL FILE ORGANIZATION
           TOTCONV=XE     HEXIDECIMAL DATA CONVERSION
  TITLE    ITEM=('WORLD''S 20 TALLEST MOUNTAINS',20)
  TITLE    ITEM=('HEXIDECIMAL DUMP OF DATA SET',22)
/*
//*
//SYSUT1   DD DSN=KC03CF7.MOUNTAIN.DATA,
//            UNIT=DISK,
//
//            DISP=(OLD,KEEP,KEEP)
//*
//*
//
```

11.4 IEBGENER

IEBGENER is a versatile utility which takes records from an input file, SYSUT1, and writes all or some of them, as they are or with editing, into an output file, SYSUT2, or into members of a PDS named in SYSUT2.

A common use of IEBGENER is to place in-stream data, such as the text of a macro or JCL procedure, into one or more members of a PDS (a macro library or procedure library). It also has other uses.

If what we want to do is modify an existing member of a PDS, e.g., replace one particular record, we do not want to use IEBGENER but IEBUPDTE instead.

JCL for IEBGENER

```
//STEP1    EXEC  PGM=IEBGENER
//SYSUT1   DD DSN=...
//SYSUT2   DD DSN=...
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
/*
```

Required DD Statements

SYSUT1 -- input file. This may be in-stream data, a sequential data set, a member of a PDS, or concatenated data sets having similar LRECL, etc.

SYSUT2 -- output file. This is a new data set or one or more new members of a new or old PDS, or it may be printed. For printing or for a new data set, supply DCB data in the DD statement.

SYSPRINT -- printed output generated by IEBGENER itself such as messages, acknowledgements, etc. LRECL=121.

SYSIN -- control statements. LRECL=80. This is normally instream data but need not be such. If all that is being done is copying all of SYSUT1 to SYSUT2 with the same BLKSIZE, all we need is SYSIN DD DUMMY.

Control Statements

GENERATE

This is used to indicate that data is being copied or edited, PDS members are being created, the number of PDS members being created and other such data.

To make sense out of the options for GENERATE, it is helpful to understand that there are two main activities available:

a. Copy a group of records from a sequential data set into new members of a PDS. In this case, it is necessary to provide a name for each member and to specify the end of the group of records.

b. Edit a record as it is being copied. This amounts to copying fields from

the input, specifying the length and location of the field, perhaps converting data from one format into another, into specified locations in the output. It is also possible to insert a literal.

Of course, we could do all of the above in one IEBGENER step.

```
      GENERATE  [,MAXNAME=n1]
                [,MAXFLDS=n2]
                [,MAXGPS=n3]
                [,MAXLITS=n4]
```

MAXNAME=n1 >= the number of member names appearing on subsequent MEMBER statements. This is required if there are any MEMBER statements at all.

MAXFLDS=n2 >= the number of FIELD parameters appearing on subsequent RECORD statements. This is required if there are any FIELD statements at all.

MAXGPS=n3 >= the number of IDENT parameters appearing on subsequent RECORD statements. This is required if there are any IDENT statements at all.

MAXLITS=n4 >= the total number of characters appearing in literals appearing in the FIELD statements of any subsequent RECORD statements. This is required if there are any literals at all.

MEMBER

This is used when we are creating members of a PDS. There must be one MEMBER statement for each PDS member being created. If no MEMBER statements are used, the output data set SYSUT2 is organized sequentially.

```
      MEMBER  NAME=name
```

Here "name" is, as usual, 1 to 8 characters long.

RECORD

Each MEMBER statement may be followed by RECORD statements until the next MEMBER statement is encountered. A RECORD statement is needed if the output is partitioned, or if editing is to be performed. We may have RECORD statements without a MEMBER statement if we are editing a sequential data set.

A RECORD statement defines a "record group", that is, a group of consecutive records to be processed identically. The last record in the group is identified by matching some part of it with a literal. If there is no RECORD statement, the entire data set is processed as one group.

```
      RECORD  [IDENT=(m1,'name',m2)]
              [FIELD=([n1,[n2 | 'literal'],[conversion],[n3])],
              [FIELD.....]
```

IDENT is used to identify the last record in a group. It matches a field in

the record, of length m1 bytes, starting at input location m2, that is, column m2, with the value specified in the literal 'name'. FIELD is used for editing. There may be numerous FIELD state-ments, as each and every byte in the output record should be speci-fied. There are two cases:

a. take a field from the input record of length n1 bytes, starting at input location n2, that is, column n2, optionally convert it as specified, and write it in the output record starting at output location n3, that is, starting in column n3,

b. write the literal specified as 'literal', optionally converted as specified, in the output record starting at output location n3, that is, starting in column n3.

The available conversions include: (these are just a sample)

(none)	no conversion
PZ	convert packed decimal numbers to unpacked decimal numbers
ZP	convert unpacked decimal numbers to packed decimal numbers

The default value of n1 is 80; if a literal is specified, n1 should be 40 or less.

The default value of n2 and n3 is 1.
The default conversion is no conversion. In this case, code two consecutive commas.

Examples

IEBGENER Control Card Example 1

```
//SYSIN DD *
  GENERATE  MAXNAME=3,MAXGPS=2
    MEMBER  NAME=MEMBER1
    RECORD  IDENT=(8,'FIRSTMEM',1)
    MEMBER  NAME=MEMBER2
    RECORD  IDENT=(8,'SECNDMEM',1)
    MEMBER  NAME=MEMBER3'
/*
```

This creates 3 members of a PDS, with names MEMBER1, MEMBER2, and MEMBER3.

The last record in MEMBER1 has 'FIRSTMEM' in its first 8 columns.

The last record in MEMBER2 has 'SECNDMEM' in its first 8 columns.

MEMBER3 contains all subsequent records from SYSUT1.

IEBGENER Control Card Example 2

```
//SYSIN DD *
  GENERATE MAXFLDS=3,MAXLITTS=8
    RECORD  FIELD=(40,1,,1),
            FIELD=(8,'NONSENSE',,41),
            FIELD=(30,41,,49)
/*
```

This edits the input record. We must have LRECL >= 70 on the input data set; the output data set should have LRECL >= 78, but if it has LRECL > 78, the columns past 78 will contain random nonsense.

Columns 1-40 of the new record will duplicate columns 1-40 of the old record.

Columns 41-48 of each new record will contain the string 'NONSENSE'.

Columns 49-78 of the new record will contain the same characters as columns 41-70 of the old record, in that order.

IEBGENER Example 1

```
//KC0nnnnA JOB ,'IEBGENER EXAMPLE 1',MSGCLASS=H
//*
//*****************************************************************
//*                                                               *
//*  THIS JOB CREATES A DATA SET AND THEN PRINTS IT.              *
//*                                                               *
//*****************************************************************
//*
//*
//GENERATE EXEC PGM=IEBGENER
//*
//*
//*****************************************************************
//*                                                               *
//*  'GENERATE' INVOKES IEBGENER TO CREATE A SEQUENTIAL FILE      *
//*  FROM AN INSTREAM DATA SET:                                   *
//*                                                               *
//*  CUSTOMER FILE FOR BANKING SYSTEM                             *
//*                                                               *
//*  INPUT FORMAT:                                                *
//*                                                               *
//*  COLUMNS    CONTENTS                                          *
//*                                                               *
//*  01-35      CUSTOMER NAME                                     *
//*  36-43      ACCOUNT NUMBER                                    *
//*  44-45      TRANSACTION TYPE                                  *
//*  46-60      TRANSACTION AMOUNT                                *
//*  61-75      CURRENT ACCOUNT BALANCE                           *
//*                                                               *
//*  OUTPUT FORMAT:                                               *
```

```
//*                                                                     *
//*  COLUMNS    CONTENTS                                                *
//*                                                                     *
//*  01-35      CUSTOMER NAME                                           *
//*  36-43      ACCOUNT NUMBER                                          *
//*  44-45      TRANSACTION TYPE                                        *
//*  46-53      TRANSACTION AMOUNT       (PACKED DECIMAL)               *
//*  54-61      CURRENT ACCOUNT BALANCE  (PACKED DECIMAL)               *
//*                                                                     *
//*  DD NAMES:                                                          *
//*                                                                     *
//*  SYSPRINT   MESSAGES FROM THE UTILITY  (OUTPUT)                     *
//*  SYSIN      CONTROL STATEMENTS         (INPUT)                      *
//*  SYSUT2     THE CREATED DATA SET       (OUTPUT)                     *
//*  SYSUT1     THE INSTREAM DATA SET      (INPUT)                      *
//*                                                                     *
//*********************************************************************
//*
//*
//SYSUT1   DD DSN=KC0nnnn.INPUT.BANK.DATA,
//            UNIT=DISK,
//
//            DISP=(OLD,KEEP,KEEP)
//*
//SYSUT2   DD DSN=KC03CF7.BANK.DATA,
//            UNIT=DISK,
//
//            DISP=(NEW,KEEP,DELETE),
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=FB,LRECL=61,BLKSIZE=610)
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  GENERATE MAXFLDS=5
  RECORD   FIELD=(35,1,,1),
           FIELD=(8,36,,36),
           FIELD=(2,44,,44),
           FIELD=(15,46,ZP,46),
           FIELD=(15,54,ZP,54)
/*
//*
//*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*
//*********************************************************************
//*                                                                     *
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT THE CREATED DATA SET.          *
//*                                                                     *
```

```
//*****************************************************************
//*
//*
//SYSUT1   DD DSN=KC0nnnn.BANK.DATA,
//            UNIT=DISK,
//
//            DISP=(OLD,KEEP,KEEP)
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  PRINT    TYPORG=PS,     PHYSICAL SEQUENTIAL FILE ORGANIZATION
           CNTRL=2,       DOUBLE SPACING
           MAXFLDS=5      FOUR FIELD OPERANDS ON RECORD OPERATION
  TITLE    ITEM=('TEST DATA FOR BANKING SYSTEM',23)
  TITLE    ITEM=('CUSTOMER NAME',13),
           ITEM=('ACCT NUM',38),
           ITEM=('TRAN',50),
           ITEM=('TRAN AMOUNT',58),
           ITEM=('ACCT BALANCE',76)
  RECORD   FIELD=(35,1,,1),
           FIELD=(8,36,,39),
           FIELD=(2,44,,51),
           FIELD=(8,46,PZ,56),
           FIELD=(8,54,PZ,74)
/*
//
```

IEBGENER Example 2

```
//KC0nnnnA JOB ,'IEBGENER EXAMPLE 2',MSGCLASS=H
//*
//*****************************************************************
//*                                                               *
//*  THIS JOB CREATES A PARTITIONED DATA SET WITH TWO MEMBERS     *
//*  AND THEN PRINTS THE TWO MEMBERS.                             *
//*                                                               *
//*****************************************************************
//*
//*
//GENERATE EXEC PGM=IEBGENER
//*
//*
//*****************************************************************
//*                                                               *
//*  'GENERATE' INVOKES IEBGENER TO CREATE A SEQUENTIAL FILE      *
//*  FROM AN INSTREAM DATA SET:                                   *
//*                                                               *
//*  CUSTOMER FILE FOR BANKING SYSTEM                             *
```

```
//*                                                                   *
//*  UNPACKED-FORMAT:                                                 *
//*                                                                   *
//*  COLUMNS    CONTENTS                                              *
//*                                                                   *
//*  01-35      CUSTOMER NAME                                         *
//*  36-43      ACCOUNT NUMBER                                        *
//*  44-45      TRANSACTION TYPE                                      *
//*  46-60      TRANSACTION AMOUNT                                    *
//*  61-75      CURRENT ACCOUNT BALANCE                               *
//*                                                                   *
//*  PACKED FORMAT:                                                   *
//*                                                                   *
//*  COLUMNS    CONTENTS                                              *
//*                                                                   *
//*  01-35      CUSTOMER NAME                                         *
//*  36-43      ACCOUNT NUMBER                                        *
//*  44-45      TRANSACTION TYPE                                      *
//*  46-53      TRANSACTION AMOUNT       (PACKED DECIMAL)             *
//*  54-61      CURRENT ACCOUNT BALANCE  (PACKED DECIMAL)             *
//*                                                                   *
//*  DD NAMES:                                                        *
//*                                                                   *
//*  SYSPRINT   MESSAGES FROM THE UTILITY (OUTPUT)                    *
//*  SYSIN      CONTROL STATEMENTS        (INPUT)                     *
//*  SYSUT2     THE CREATED DATA SET      (OUTPUT)                    *
//*  SYSUT1     THE INSTREAM DATA SET     (INPUT)                     *
//*                                                                   *
//*******************************************************************
//*
//*
//SYSUT1   DD *
JOSEPH JAQUARD                    1111WD000000000050000000000000050000
CHARLES BABBAGE                   1212DP000000000025000000000000123456
BLAISE PASCAL                     1313DP000000000010000000000004532100
AUGUSTA ADA BYRON                 1414WD000000000005000000000000007500
HERMAN HOLLERITH                  1515WD000000001500000000000002879913
THOMAS J.  WATSON, SR.            1616DP000000002129913000000000000100
HOWARD AIKEN                      1717WD000000000047777000000005800917
DR.  KINSATP .J DRAHCIR           1818DP000000000012345000000000100013
CLIFFORD BERRY                    1919WD000000006879022000000000023789
DR.  JOHN W.  MAUCHLY             2121DP000000001124989000000088125609
END MEM1
J.  PRESPER ECKERT, JR.           2222DP000000000050000000000000150000
DR.  JOHN VON NEUMANN             2323WD000000000025000000000000235609
SEYMOUR CRAY                      2424DP000000000010000000000000120000
ANNIE GLIDDEN                     2525WD000000000005000000000000100000
SWEN PARSON                       2626DP000000000012457000000000967541
GRANT TOWERS                      2727WD000000002246823000000000000109
STEVEN TOWERS                     2828DP000000000045000000000000120099
DOUGLAS HALL                      2929DP000000000120000000000001326000
```

```
GILBERT HALL                        3131WD00000000034560000000998003487
MONTGOMERY HALL                     3232WD00000000009798000000000067564
/*
//*
//SYSUT2   DD DSN=&&BANKLIB,
//            DISP=(NEW,PASS,DELETE),
//            SPACE=(TRK,(1,1,1)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  GENERATE MAXNAME=2,MAXGPS=1,MAXFLDS=10
  MEMBER   NAME=BANKPACK
  RECORD   IDENT=(8,'END MEM1',1),
           FIELD=(35,1,,1),
           FIELD=(8,36,,36),
           FIELD=(2,44,,44),
           FIELD=(15,46,ZP,46),
           FIELD=(15,54,ZP,54)
  MEMBER   NAME=BANKUNPK
  RECORD   FIELD=(35,1,,1),
           FIELD=(8,36,,36),
           FIELD=(2,44,,44),
           FIELD=(15,46,,46),
           FIELD=(15,61,,61)
/*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*
//*****************************************************************
//*                                                               *
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT ONE PDS MEMBER.          *
//*                                                               *
//*****************************************************************
//*
//*
//SYSUT1   DD DSN=&&BANKLIB,
//            DISP=(OLD,PASS,DELETE)
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  PRINT    TYPORG=PO,     PARTITIONED DATA SET ORGANIZATION
           TOTCONV=XE,    HEXIDECIMAL DATA CONVERSION
           MAXNAME=1      ONE MEMBER STATEMENT
  TITLE    ITEM=('PACKED TEST DATA FOR BANKING SYSTEM',16)
  TITLE    ITEM=('HEXIDECIMAL DUMP OF DATA SET',19)
```

```
  MEMBER   NAME=BANKPACK
/*
//*
//PRINTIT  EXEC PGM=IEBPTPCH
//*
//*****************************************************************
//*  'PRINTIT' INVOKES IEBPTPCH TO PRINT ONE PDS MEMBER.         *
//*****************************************************************
//*
//*
//SYSUT1   DD DSN=&&BANKLIB,
//            DISP=(OLD,PASS,DELETE)
//*
//SYSUT2   DD SYSOUT=*
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSIN    DD *
  PRINT    TYPORG=PO,     PARTITIONED DATA SET ORGANIZATION
           TOTCONV=XE,    HEXIDECIMAL DATA CONVERSION
           MAXNAME=1      ONE MEMBER STATEMENT
  TITLE    ITEM=('UNPACKED TEST DATA FOR BANKING SYSTEM',17)
  TITLE    ITEM=('HEXIDECIMAL DUMP OF DATA SET',21)
  MEMBER   NAME=BANKUNPK
/*
//*
//*
//
```

11.5 IEBCOPY

IEBCOPY is a utility to copy or merge all or part of one or more PDSs, often for the purpose of making a backup copy. Members can be renamed or replaced with identically named members. A PDS can be copied to a sequential data set. A PDS can be compressed, releasing unused space.

JCL for IEBCOPY

```
//KC0nnnnA JOB ,'PROGRAMMER',MSGCLASS=H
//STEP1  EXEC  PGM=IEBCOPY[,REGION=...][,PARM=...]
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=...
//SYSUT2   DD DSN=...
//SYSIN    DD *
/*
//*
```

As usual, we can specify a REGION on the EXEC statement. The PARM can have various values, including:

```
     LINECOUNT=nn          ¿ =number of lines to be printed on
or                           ³    each page.  The default is 60.
     LC= nn                Ù
```

Other possibilities for PARM including setting a buffer size (not recommended – Let IEBCOPY do it) and various values which substitute for the same information given in SYSIN instead.

Required DD Statements

SYSUT1 and SYSUT2 (or "anyname") are DDs which define PDSes. Normally SYSUT1 is for input and SYSUT2 is for output. There may be more one input DD and more than one output DD.

SYSUT3 and SYSUT4 are optional "spill" data sets on DASDs. These are optional and are used for overflow situations.

SYSPRINT -- printed output generated by IEBCOPY itself such as messages, acknowledgements, etc. LRECL >= 60 and <= 250. We always have DSORG=PS. By default, RECFM=FBA and LRECL=121.

SYSIN -- control statements. This is normally in-stream data with LRECL=80 but need not be such. If SYSIN DD DUMMY is used, the result is the same as using:

```
         COPY OUTDD=SYSUT2,INDD=SYSUT1
```

Control Statements (There are more not covered here.)

COPY

This parameter initiates a copy operation. It is used as follows:

```
     COPY  OUTDD=ddname1,INDD=ddname2[,LIST=YES/NO]
```

Here OUTDD and INDD name the output and input PDSes.

Instead of INDD=ddname2, we could have INDD=(ddname2,R); here the "R" means "replace"; any member of the input PDS whose name matches the name of a member of the output PDS will replace that member.

LIST=YES (the default) causes the members copied to be listed in the printed output from IEBCOPY.

COPY may be abbreviated to "C", INDD to "I" and OUTDD to "O".

SELECT

This specifies which members in the input PDS should be copied.

It is used as follows:

```
     SELECT MEMBER=MEM1
or
     SELECT MEMBER=(MEM1,MEM2)
or
     SELECT MEMBER=((MEM1,NEW1,R))
or
     SELECT MEMBER=((MEM1,,R))
```

Here the "R" indicates that a member should be renamed (if NEW1 is specified) or replaced (if NEW1 is missing).

We can follow SELECT by a list of several members to be copied, renamed or replaced.

SELECT may be abbreviated to "S" and MEMBER to "M".

EXCLUDE

This specified members which are not to be copied. It is used as follows:

```
     EXCLUDE MEMBER=MEM1
or
     EXCLUDE MEMBER=(MEM1,MEM2)
```

EXCLUDE may be abbreviated to "E" and MEMBER to "M".

IEBCOPY Examples

```
//KC0nnnnA JOB ,'IEBCOPY EXAMPLES',MSGCLASS=H
//*
//*****************************************************************
//*  THIS JOB CREATES A PARTITIONED DATA SET FROM SELECTED       *
//*  MEMBERS OF AN EXISTING PARTITIONED DATA SET, UPDATES TWO    *
```

```
//*  MEMBERS, AND THEN COMPRESSES THE PDS.                        *
//*****************************************************************
//*
//*
//COPY1    EXEC PGM=IEBCOPY
//*
//*****************************************************************
//*                                                               *
//*  'COPY1' INVOKES THE IEBCOPY UTILITY TO CREATE A PDS.         *
//*                                                               *
//*  SYSIN     CONTROL STATEMENTS FOR IEBCOPY      (INPUT)       *
//*  SYSPRINT  MESSAGES FROM IEBCOPY               (OUTPUT)      *
//*  SYS2MACS  AN EXISTING PDS                     (INPUT)       *
//*  NEWMACS   THE CREATED PDS                     (OUTPUT)      *
//*                                                               *
//*****************************************************************
//*
//SYSPRINT DD SYSOUT=*
//SYS2MACS DD DSN=KC00NIU.SYS2.MACLIB,DISP=SHR
//NEWMACS  DD DSN=&&NEWMACS,DISP=(NEW,PASS,DELETE),
//            SPACE=(TRK,(1,1,1))
//SYSIN    DD *
  COPY     INDD=SYS2MACS,OUTDD=NEWMACS
  SELECT   MEMBER=XDECI,XPRNT,XDECO,XSAVE,EQUREGS
/*
//*
//*
//*
//COPY2    EXEC PGM=IEBCOPY
//*
//*****************************************************************
//*                                                               *
//*  'COPY2' INVOKES THE IEBCOPY UTILITY TO UPDATE TWO MEMBERS.   *
//*                                                               *
//*  SYSIN     CONTROL STATEMENTS FOR IEBCOPY      (INPUT)       *
//*  SYSPRINT  MESSAGES FROM IEBCOPY               (OUTPUT)      *
//*  SYS2MACS  AN EXISTING PDS                     (INPUT)       *
//*  NEWMACS   THE UPDATED PDS                     (OUTPUT)      *
//*                                                               *
//*****************************************************************
//*
//SYSPRINT DD SYSOUT=*
//SYS2MACS DD DSN=KC00NIU.SYS2.MACLIB,DISP=SHR
//NEWMACS DD DSN=&&NEWMACS,DISP=(OLD,PASS,DELETE)
//SYSIN    DD *
  COPY     INDD=SYS2MACS,OUTDD=NEWMACS
  SELECT   MEMBER=((XSAVE,,R),(EQUREGS,,R))
/*
//*
//*
//*
```

```
//COPY3    EXEC PGM=IEBCOPY
//*
//*****************************************************************
//*                                                               *
//*  'COPY3' INVOKES THE IEBCOPY UTILITY TO COMPRESS A PDS.       *
//*                                                               *
//*  SYSIN     CONTROL STATEMENTS FOR IEBCOPY      (INPUT)       *
//*  SYSPRINT  MESSAGES FROM IEBCOPY               (OUTPUT)      *
//*  NEWMACS   THE EXISTING PDS TO BE COMPRESSED   (I-O)         *
//*                                                               *
//*****************************************************************
//*
//SYSPRINT DD SYSOUT=*
//*
//NEWMACS  DD DSN=&&NEWMACS,DISP=(OLD,PASS,DELETE)
//*
//SYSIN    DD *
  COPY     INDD=NEWMACS,OUTDD=NEWMACS
/*
//
```

11.6 IEHLIST

IEHLIST is a utility used to list members in a PDS. It is not used much any more.

IEHLIST Example

```
//KC0nnnnA JOB ,'PROGRAMMER',MSGCLASS=H
//*****************************************************************
//*                                                               *
//*  This job uses IEHLIST to list the members of a PDS,          *
//*  EXAMPLE.LIB.  For the sake of illustration, let's say it is  *
//*  on ACA301.                                                   *
//*                                                               *
//*  DD Name             File Description                         *
//*  SYSPRINT    Output:  Standard output for IEHLIST messages.   *
//*  ANY1       Utility:  Placeholder.                            *
//*  SYSIN        Input:  Control statements.                     *
//*                                                               *
//*  In this example, the control statement used is LISTPDS.      *
//*                                                               *
//*  We could have several LISTPDS statements; each would have    *
//*  an "ANYNAME" DD to match it.  Each LISTPDS has two parameters, *
//*  DSNAME and VOL=device=serial.  If the PDS resides on the     *
//*  system residence volume (SYSRES), VOL is not needed.  If the *
//*  PDS is a library of load modules, LISTPDS may also have the  *
//*  parameter FORMAT; alternatively, or as the default, it may   *
//*  have the parameter DUMP.                                     *
//*                                                               *
//*  IEHLIST can also be used to list the catalog of a volume.    *
//*                                                               *
//*  The device listed on VOL= is usually DISK.  The serial number  *
//*  is the serial number of the VOL=SER for EXAMPLE.LIB, and     *
//*  ANY1 is on the same VOL=SER.  Since we cannot specify which  *
//*  Volume to create a dataset on, We must know beforehand which *
//*  volume the PDS is on, to successfully run IEHLIST.           *
//*                                                               *
//*****************************************************************
//*
//STEP   EXEC  PGM=IEHLIST
//SYSPRINT DD SYSOUT=*
//ANY1     DD SPACE=(TRK,0),DISP=OLD
//SYSIN    DD *
     LISTPDS DSNAME=EXAMPLE.LIB,VOL=3390=volume-name
/*
//
```

Note

Because `IEHLIST` requires that you know the `VOL=SER=` for the PDS and Marist uses SMS, the user must first find the data set and its Marist volume using TSO/ISPF Utilities `DSLIST` Option and then run the `IEHLIST` to print the PDS's directory information.