UNIT 5

Utilities

Utilities

- Objectives
- OS/390 Utilities
- Utility Classification
- Utility Control Statement Syntax
- IEBGENER Utility
- IEBPTPCH Utility
- IEHLIST Utility
- IEBCOPY Utility
- IEHPROGM Utility
- IEBUPDTE Utility
- IDCAMS Utility
- SORT Utility

Objectives

- Understand the concept of a Utility
- Know how to code different Utilities
- Understand the functions of different Utilities

OS/390 Utilities

- Utilities are IBM-supplied programs that are intended to perform certain routine and frequently occurring tasks in the OS/390 environment.
- Utilities are used in DASD, tape drives, print and punch operations.
- Utilities are used to allocate, update, delete, catalog and uncatalog data sets, and also to list the contents of VTOC (Volume Table of Contents).

Utility Classification

Utilities are broadly classified into two different categories:

- Data set Utilities (prefixed with IEB)
- System Utilities (prefixed with IEH)

Data set utilities are used to copy, print, update, reorganize, and compare data at the dataset and/or record level.

System utilities are used to list VTOC information, copy, delete, catalog and uncatalog datasets, to write tape labels and to add or delete dataset passwords.

Utility Control Statement (SYSIN)

During execution, utilities open and read the dataset described in the DD statement SYSIN. The Control statement parameters are given in this DD statement.

If no control statements are required then you have to use the DUMMY parameter of the DD statement, since the program expects some kind of input (in the form of control statement).

The format in which the SYSIN DD statement will appear is:

```
//SYSIN   DD   *
  ----Control statements-----
/*
```

Another way to code the SYSIN is to code the control statements in a member of a PDS. Then the SYSIN DD would appear as follows:

```
//SYSIN   DD   DSN=MAINUSR.CTL.LIB(MEM),DISP=SHR
```

Utility Control Statement Syntax

The general format of a Utility Control Statement is :

[label] operation operands comments

Where:

Field	
label	is optional.
operation	is required.
operands	is required.
comments	is optional.

Utility control statements must be coded in columns 1 thru 71.

If a statement exceeds column 71, then

1. Break the statement in column 71 or after a comma.
2. Code a non-blank character in column 72.
3. Continue the statement in column 16 of the following line.

The IEBGENER Utility

IEBGENER is a dataset utility used to create, copy, or print sequential data sets

Example:

```
//JOBNAME  JOB  ACCT,'APARNASANDEEP'
//STEP1    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   DSN=MAINUSR.SEQ.INPUT,DISP=SHR
//SYSUT2   DD   DSN=ABC.SEQ.OUTPUT,
//               DISP=(,CATLG,DELETE),
//               SPACE=(TRK,(1,1)),VOL=SER=LP1WK2,
//               RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD   DUMMY
```

In the above example, the dataset MAINUSR.SEQ.INPUT is being copied to a new file called ABC.SEQ.OUTPUT.

- The EXEC statement specifies the program to be executed (IEBGENER).
- The SYSPRINT DD statement defines the message dataset.
- The SYSUT1 DD statement defines the input dataset.
- The SYSUT2 DD statement defines the output dataset (can have multiple SYSUTn, where n should be 1,2,3…).
- The SYSIN DD statement defines the control dataset. This is where IEBGENER looks for any utility control statements. When DUMMY is specified, there are no control statements being used.

The IEBGENER Utility

This example shows how IEBGENER can be used for copying or printing selected portions of datasets. In this case, selected records from MAINUSR.ABC.REC will be printed.

Example:

```
//JOBNAME  JOB  ACCT,'APARNASANDEEP'
//STEP1    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   DSN=MAINUSR.ABC.REC
//SYSUT2   DD   SYSOUT=*
//SYSIN    DD   *
             GENERATE  MAXFLDS=2
             RECORD FIELD=(10,20,,1),FIELD=(10,1,,15)
/*
//
```

This step executes IEBGENER to create the SYSUT2 output data and route it to a print class.

- 'SYSIN DD *' indicates that instream records or control statements follow this statement.
- The GENERATE statement specifies that editing of the input data is to be performed.
- MAXFLDS=2 indicates that no more than 2 fields will be described on the subsequent RECORD statement.
- The RECORD statement describes the input and output fields thru the FIELD parameter.
- Each FIELD parameter specifies the field's length, its location in the input record, what type of conversion is to be done on the field, and where it should be placed in the output record. The format is: FIELD=(LENGTH OF FIELD,POSITION IN INPUT, CONVERSION, POSITION IN OUTPUT)

  The FIELD parameters in the above example state to:
  - move the 10 bytes starting in position 20 of the input record to the 10 bytes starting in 1 in the output record;
  - and to move the 10 bytes starting in position 1 of the input record to the 10 bytes starting in 15 in the output record

The IEBPTPCH Utility

IEBPTPCH is used to print or punch all or selected portions of datasets. Editing can be done on the data to format the output.

Example:

```
//STEP1     EXEC PGM=IEBPTPCH
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DSN=MAINUSR.ABC.REC,DISP=SHR
//SYSUT2    DD   SYSOUT=*
//SYSIN     DD   *
          PRINT  TYPROG=PS,MAXFLDS=2
          TITLE ITEM=('EMPLOYEES PROFILE',27)
          TITLEITEM=('NAME       ADDRESS',15)
          RECORD FIELD=(8,2,,10),FIELD=(5,10,,20)
```

- The PRINT control statement specifies that the dataset has an organization of physical sequential and that a maximum of two fields will be printed on output.

- The output titles are specified on the TITLE control statements.

- 'EMPLOYEES PROFILE' will be placed position 27 of the output file
- 'NAME' and 'ADDRESS' will be placed on the next title line, beginning in position 15

- The RECORD statement describes the input and output fields thru the FIELD parameter.

- Each FIELD parameter specifies the field's length, its location in the input record, what type of conversion is to be done on the field, and where it should be placed in the output record. The format is: FIELD=(LENGTH OF FIELD,POSITION IN INPUT, CONVERSION, POSITION IN OUTPUT)

- The FIELD parameters in the above example state to:
- move the 8 bytes starting in position 2 of the input record to the 8 bytes starting in 10 in the output record;
- and to move the 5 bytes starting in position 10 of the input record to the 5 bytes starting in 20 in the output record

The IEHLIST Utility

The IEHLIST utility is used to

- list entries in a DASD VTOC (Volume Table of Contents)
- list entries in a PDS Directory.
- list entries in a system catalog

Example 1:

```
//STEP1     EXEC  PGM=IEHLIST
//SYSPRINT  DD    SYSOUT=*
//DD1       DD    DISP=OLD,UNIT=SYSDA,VOL=SER=ABC
//DD2       DD    DISP=OLD,UNIT=SYSDA,VOL=SER=DEF
//SYSIN     DD    *
       LISTVTOC  FORMAT,VOL=SYSDA=ABC
       LISTVTOC  FORMAT,VOL=SYSDA=DEF                          X
                 DSNAME=(MTPL.FILE1,MTPL.FILE2)
/*
//
```

The above example uses IEHLIST to print two VTOC listings:

- The IEHLIST looks for utility control statements coded below the SYSIN DD statements:
- The first LISTVTOC control statement requests an formatted (FORMAT) VTOC listing for pack ABC. This includes DSCB and space allocation information. If FORMAT is omitted, an abbreviated version is listed.
- The second LISTVTOC control statement requests a formatted VTOC listing for two datasets: MTPL.FILE1 and MTPL.FILE2.

Example 2:

```
//STEP1     EXEC PGM=IEHLIST
//SYSPRINT  DD   SYSOUT=*
//DD1       DD   DISP=OLD,UNIT=SYSDA,VOL=SER=ABC
//SYSIN     DD   *
        LISTPDS DSNAME=MTPL.FILE,VOL=SYSDA=ABC
/*
//
```

The above example uses IEHLIST to list entries in a PDS directory.

- The LISTPDS control statement requests a listing of the directory for the PDS, MTPL.FILE.

NOTE: DSNAME cannot be abbreviated as DSN on a control statement

The IEBCOPY Utility

The IEBCOPY is used to copy members of partitioned datasets.

The COPY statement identifies the input and output files by referring to their DDNAMEs in the JCL. The format is:

```
COPY   OUTDD=output-ddname,INDD=input-ddname
```

The SELECT statement identifies the members of the PDS to be copied. The format is:

SELECT MEMBER=NAME (to specify a single member)

-or- **SELECT MEMBER=(NAME,NAME,NAME)** (to specify multiple members)

-or- **SELECT MEMBER=(old,new,R)** (to copy a member and change its name)

The EXCLUDE statement indicates that all members should be copied except those specified in the EXCLUDE statement. The format is:

EXCLUDE MEMBER=NAME (to specify a single member)

-or- **EXCLUDE MEMBER=(NAME,NAME,NAME)** (to specify multiple members)

Copying an Entire PDS

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//IN         DD   DSN=MTPL.FILE1,DISP=SHR
//OUT        DD   DSN=MTPL.FILE2,DISP=SHR
//SYSIN      DD   *
     COPY   OUTDD=OUT,INDD=IN
/*
```

The above example copies all of the members from the PDS, '**MTPL.FILE1**' to an existing PDS, '**MTPL.FILE2**'.

- The IN and OUT DD statements define data sets to be used by IEBCOPY.
- The COPY control statement specifies the input and output DDNAMES.
- No SELECT or EXCLUDE statements were used.

Copying Specific Members

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//IN         DD   DSN=MTPL.FILE1,DISP=SHR
//OUT        DD   DSN=MTPL.FILE2,DISP=SHR
//SYSIN      DD   *
     COPY   OUTDD=OUT,INDD=IN
     SELECT MEMBER=ALLOCATE
     SELECT MEMBER=((PROD,TEST,R))
/*
```

The above example copies the member called "ALLOCATE" from the PDS, '**MTPL.FILE1**' to an existing PDS, '**MTPL.FILE2**'.

- The SELECT control statement specifies the member ALLOCATE is to be copied from the input to the output dataset.

Copying Multiple Specific Members

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//IN         DD   DSN=MTPL.FILE1,DISP=SHR
//OUT        DD   DSN=MTPL.FILE2,DISP=SHR
//SYSIN      DD   *
     COPY   OUTDD=OUT,INDD=IN
     SELECT MEMBER=(FILE1,FILE2,FILE3)
/*
```

The above example copies selected members from the PDS, '**MTPL.FILE1**' to an existing PDS, '**MTPL.FILE2**'.

- The SELECT control statement specifies the members: FILE1, FILE2 and FILE3 to be copied.

Copying and Renaming Specific Members

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//IN         DD   DSN=MTPL.FILE1,DISP=SHR
//OUT        DD   DSN=MTPL.FILE2,DISP=SHR
//SYSIN      DD   *
     COPY   OUTDD=OUT,INDD=IN
     SELECT MEMBER=(JOBA,(PROD,TEST,R))
/*
```

The above example copies the member called "PROD" from the PDS, '**MTPL.FILE1**' to an existing PDS, '**MTPL.FILE2**'.

The SELECT control statement specifies:

- Copy the member JOBA
- the member PROD is to be copied in the following manner
  - rename PROD to TEST,
  - copy the renamed member TEST to the output dataset,
  - if a member by that name exists in the output dataset replace it.

Copying Using EXCLUDE

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//IN         DD   DSN=MTPL.FILE1,DISP=SHR
//OUT        DD   DSN=MTPL.FILE2,DISP=SHR
//SYSIN      DD   *
     COPY   OUTDD=OUT,INDD=IN
     EXCLUDE MEMBER=ALLOCATE
/*
```

The above example copies all members '**MTPL.FILE1**' except the member 'ALLOCATE'

Compressing Data Sets

```
//MODAL2     JOB  '0.2AMIP',…….
//STEP1      EXEC PGM=IEBCOPY
//SYSPRINT   DD   SYSOUT=*
//INPDS      DD   DSN=MTPL.FILE1,DISP=SHR
//SYSUT3     DD   UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSUT4     DD   UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSIN      DD   *
     COPY INDD=INPDS,OUTDD=INPDS
/*
```

The above example compresses the library '**MTPL.FILE1**'.
Notice that the same DD name is specified in both the INDD and OUTDD parameters.

The IEHPROGM Utility

The IEHPROGM is used to:

1. Scratch (delete) a dataset
2. Rename a member of a PDS
3. Catalog or uncatalog datasets

Example 1:

```
//STEP1     EXEC PGM=IEHPROGM
//SYSPRINT  DD   SYSOUT=*
//NUM1      DD    UNIT=SYSDA,VOL=SER=ABC,DISP=OLD
//SYSIN     DD    *
      SCRATCH MEMBER=ALLOCATE,DSNAME=MTPL.FILE,            X
      VOL=SYSDA=ABC
      RENAME    MEMBER=XYZ,DSNAME=MTPL.FILE1,              X
      VOL=SYSDA=ABC,NEWNAME=PQR
/*
//
```

In the above example

- The SCRATCH statement tells IEHPROGM is used to scratch the member ALLOCATE in the PDS, MTPL.FILE.
- The RENAME control Statement tells IEHPROGM to rename member XYZ to PQR in the PDS, MTPL.FILE1.

Example 2:

```
//IEHPROGM2      JOB A123,'MAHESH',……..
//STEP1    EXEC  PGM=IEHPROGM
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD     UNIT=3390,VOL=SER=LP2WK1
//SYSIN    DD     *
      UNCATLG    DSNAME=ABC.FILE,VOL=SER=LP2WK1,UNIT=3390
/*
//
```

In the above example:

- The UNCATLG statement tells IEHPROGM to uncatalog the dataset ABC.FILE
- Use CATALOG to catalog a dataset

The IEBUPDTE Utility

The IEBUPDTE utility is used to create, update and modify sequential datasets and members of partitioned datasets.

Example

```
//STEP1      EXEC PGM=IEBUPDTE
//SYSPRINT   DD   SYSOUT=*
//SYSUT1     DD   DSN=MTPL.MYPDS,VOL=SER=ABC,UNIT=SYSDA,DISP=OLD
//SYSIN      DD   *
. /   CHANGE NAME=MEM,UPDATE=INPLACE
           JAY    HARI  00000050
           MARY   CHRISTI     00000070
. /   ENDUP
/*
```

When a PDS member is changed and then replaced or added to a PDS, it normally goes in the available space after the last member in the PDS. If several records in a member are replaced by an equal number of records, IEBUPDTE can update the member without changing its address in the PDS. This is called Update in Place.

- In the above example, the PDS MTPL.MYPDS, contains a member named MEM, which contains two records with sequence numbers (00000050 and 00000070) in columns 73 thru 80.When the IEBUPDTE job is executed, those two records will be replaced by the two records in the SYSIN input stream. This is an Update in Place.

Before:

```
ELIZABETH  HENRY                00000030
THOMAS     JOHNSON              00000040
RICO       BROWN                00000050
TIMOTHY    SIMMS                00000060
MARY       BAKER                00000070
HARRIET    WILLIAMS             00000080
```

After:

```
ELIZABETH  HENRY                00000030
THOMAS     JOHNSON              00000040
JAY         HARI                00000050
TIMOTHY    SIMMS                00000060
MARY       CHRISTI              00000070
HARRIET    WILLIAMS             00000080
```

The IEFBR14 Utility

This utility is called a dummy utility, since its basic function is to do what the disposition parameter of the DD says.

To Uncatalog a Dataset

```
//UNCATLOG  JOB  A123,'SUSAN JOHN'
//STEP1     EXEC PGM=IEFBR14
//DD1       DD   DSN=MAINUSR.COBOL.FILE1,DISP=(OLD,DELETE)
```

To Delete The Dataset

```
//DELETE1  JOB    A123,' 'SUSAN JOHN'
//STEP1    EXEC   PGM=IEFBR14
//DD1       DD    DSN=MAINUSR.COBOL.FILE2,DISP=(OLD,DELETE)
```

To Catalog a Dataset

```
//CATALOG  JOB   A123,' 'SUSAN JOHN'
//STEP1    EXEC  PGM=IEFBR14
//DD1       DD   DSN=MAINUSR.COBOL.FILE3,DISP=(NEW,CATLG)
```

To Create A Dataset

```
//CREATE   JOB     A123, 'SUSAN JOHN'
//STEP3    EXEC    PGM=IEFBR14
//SYSIN DD SYSOUT=*
//DD1       DD     DSN=MAINUSR.COBOL.FILE4,DISP=(NEW,KEEP),
//                       UNIT=SYSDA,SPACE(TRK,(4,2)),
//                       DCB=(LRECL=80,RECFM=FB,BLKSIZE=800),
//                       VOL=SER=LP2WK1
//SYSIN DD DUMMY
```

The IDCAMS Utility

IDCAMS is a special utility to create, delete, copy and print the contents of VSAM and non-VSAM datasets.

Example:

```
//JOBNAME   JOB  (ACCT),'SUSAN JOHN'
//STEPNAME  EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD  *
            Functional-Commands OR Control Statements
/*          (terminator)
//
```

Example:

```
//JOBNAME   JOB (ACCT),'SUSAN JOHN'
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
      DELETE MTPL.SEQ.DATA
/*
//
```

In the above example when the step executes, IDCAMS deletes the non-VSAM dataset MTPL.SEQ.DATA.

Functions coded on IDCAMS Utility
The IDCAMS Utility can be executed for the following function on VSAM datasets.

- Define them
- Load records into them
- Print them

The IDCAMS Utility (Continued…)

Example:

```
//KSDLOAD1     JOB  (A123),'SUSAN JOHN',
//STEP1        EXEC PGM=IDCAMS
//SYSPRINT     DD   SYSOUT=*
//DDIN         DD   DSN=FILE1.TEST,DISP=SHR
//DDOUT        DD   DSN=VSAM1.KSDS.CLUSTER,DISP=OLD
//SYSIN        DD   *
     REPRO            -
     INFILE(DDIN)     -
     OUTFILE(DDOUT)
//*
//
```

Alter Command to alter a PDS

```
//ALTER1   JOB  23,'SUSAN JOHN'
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
        ALTER   -
        MAINUSR.COBOL.LOADLIB (PROGRAM1)                -
        NEWNAME(MAINUSR.COBOL.LOADLIB (SAMPLE1))
/*
//
```

Here the MAINUSR.COBOL.LOADLIB (PROGRAM1) is changed to MAINUSR.COBOL.LOADLIB (SAMPLE1)

The SORT Utility

The SORT utility is used is sort the contents of a dataset depending on a key called the sort field. The sorted output is then written to another dataset

SORT is an alias or alternate name for ICEMAN. PGM=SORT or PGM=ICEMAN on the EXEC statement will invoke DFSORT program which is used for sorting an input dataset.

The SORT control statement is specified in the SYSIN DD statement.

Example:

```
//STEP1    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=MAINUSR.SEQ1.INPUT,DISP=OLD
//SORTOUT  DD  DSN=MAINUSR.SEQ2.OUTPUT,DISP=OLD
//SYSIN    DD  *
         SORT FIELDS=(21,2,CH,A)
/*
```

The above example will sort the records of the input dataset specified in the SORTIN DD statement based on the field specified in the control statement of the SYSIN DD. The sorted dataset is copied to the output dataset specified in the SORTOUT DD statement.

- The EXEC statement invokes the program.
- The SORTIN DD statement defines the input data set.
- The SORTOUT DD statement defines the output data set.
- The SYSIN DD statement defines the control data set.
- The SORT control statement specifies the position, length, format and order of sort. The above example sorts the records in ascending (A) order, using the 2 bytes (2) of character data (CH) starting in location 21.

The SORT Utility (Continued…)

Example 2

```
//SORT1    JOB   (A123),'SUSAN JOHN',NOTIFY=&USERID
//STEP1    EXEC  PGM=SORT
//SYSOUT   DD    SYSOUT=A
//SYSPRINT DD    SYSOUT=A
//SORTIN   DD    DSN=MAINUSR.ADDRESS.BOOK1,DISP=SHR
//SORTOUT  DD    DSN=MAIN006.ADDRESS.BOOK2,
//                    DISP=(NEW,CATLG,DELETE),
//                    UNIT=SYSDA,
//                    SPACE=(CYCL,(2,1),RLSE),VOL=SER=LP2WK1
//                    DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(20,10),RLSE)
//SYSIN     DD        *
     SORT   FIELDS=(2,3,CH,A)
/*
//
```

The above example sorts ascending on the 3-byte string which starts in position 2.

Assume MAINUSR.ADDRESS.BOOK1 in the input file has three records

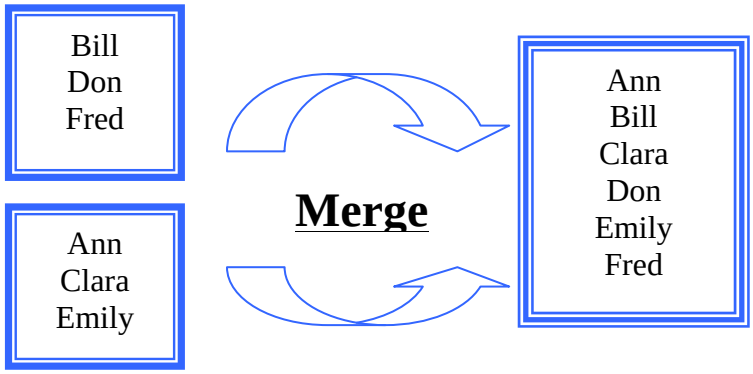
- 089134 - Ms. Patti Smith,Nevada
- 012345 - Mr.John Henley,Californa
- 042345 - Mr. Abraham Scott,NewYork

After successful completion the output file will look into this.

- 012345 – Mr. John Henley,California
- 042345 - Mr. Abraham Scott,NewYork
- 089134 - Ms. Patti Smith,Nevada

The MERGE UTILITY

The MERGE utility is used to combine two or more sorted files into a single sorted file



- This process takes records from up to 16 sorted data sets and combines them in a single sorted output data set.
- Each of the input data sets must be previously sorted in the same sequence before the merge
- In the above example two sorted data sets are merged into a single sorted data set.

Example

```
//STEP1    EXEC PGM=ICEMAN
//SYSOUT   DD   SYSOUT=*
//SORTIN   DD   DSN=MTPL.SEQ1,DISP=OLD
//SORTIN2  DD   DSN=MTPL.SEQ2,DISP=OLD
//SORTOUT  DD   DSN=MTPL.SEQ3,DISP=OLD
//SYSIN         DD *
           MERGE FIELDS =(21,2,CH,A)
/*
```

- The merge control statement format is similar to that of the sort. Input data sets must already be sorted in the same sequence.
- If all fields have the same format, then the merge control statement can be written in the form:

MERGE FIELDS =(21,2,CH,A)

Unit 5 Exercises

1. **The _____ utility can be used to create, copy or print sequential datasets.**

2. **The _____ utility is used to allocate and delete VSAM, non-VSAM datasets.**

3. **The _____ utility is used to copy the contents from one PDS to another.**

4. **The _____ DD name is used to code the control statement(s) for a utility program.**

5. **The _____ utility is used to print the contents of a dataset.**

6. **The _____ utility is used to list the contents of a DASD VTOC.**

7. **The sort utility uses the _____ program to sort an input dataset.**

Unit 5 Lab Exercises

Logon to TSO/ISPF and perform the following exercises. Wherever you see "userid" in lower case, substitute your valid security userid.

Create a Physical Sequential Dataset

1. In your PDS called 'userid.JCL.CNTL', create a new member called JOBTEST5.

2. Write a JOB Statement using the following criteria.
   - Job name - Your Userid & an additional character
   - Account field - your valid account number
   - Programmers name - Userid
   - Job Log & system messages - Send to print class X
   - Messages should be sent to the TSO user when JOB processing is completed.
   - Maximum execution time 10 minutes.

IEBGENER

Use the JCL listed below as a guide to this lab.

3. Add a step called //STEP1 to execute the program IEBGENER.

4. Include one DD statement called SYSPRINT. It should put the SYSOUT on the same Print class as indicated in the JOB statement.

5. Include one DD statement called SYSUT1 referring to the member JOBTEST2 on the library 'userid.JCL.CNTL'

6. Include one DD statement named SYSUT2. It should allocate a new dataset called userid.JCL.LAB5A that JOBTEST2 will be copied to.

7. Include one DD statement named SYSIN for a dataset that does not exist.

8. Submit your job and review/debug the results.

```
//STEP1    EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   DSN=userid.JCL.CNTL(JOBTEST2),DISP=SHR
//SYSUT2   DD   DSN=userid.JCL.LAB5A,
//               DISP=(,CATLG,DELETE),
//               SPACE=(TRK,(1,1)),
//               RECFM=FB,LRECL=80,UNIT=SYSDA
//SYSIN    DD   DUMMY
```

IEBCOPY

Use the JCL listed below as a guide to this lab.

9. In your PDS called ‘userid.JCL.CNTL’, create a new member called JOBTEST6.

10. Copy the job card from JOBTEST5, and rename the job to userid6.

11. Add a step called //STEP1 to execute the program IEBCOPY.

12. For this step:

    a. Copy the member ‘JOBTEST2’ from userid.JCL.CNTL to a new PDS called userid.JCL.CNTL1
    b. Rename ‘JOBTEST2’ to ‘COPY1 during the copy.

```
//STEP1    EXEC PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=*
//IN       DD   DSN=userid.JCL.CNTL,DISP=SHR
//OUT      DD   DSN=userid.JCL.CNTL1,DISP=(NEW,CATLG),
//         SPACE=(TRK,(2,3,1)RLSE),VOL=SER=SYSDA
//SYSIN    DD   *
     COPY   OUTDD=OUT,INDD=IN
     SELECT MEMBER=((JOBTEST2,COPY1,R))
```