AAEC/E267

AAEC/E267

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

AEUPDATE - AN EDITOR DESIGNED TO ALLOW SEQUENTIAL AND PARTITIONED DATA SETS TO BE UPDATED

by

I.L. JOHNSTONE

August 1973

ISBN 0 642 99586 9

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AEUPDATE - AN EDITOR DESIGNED TO ALLOW SEQUENTIAL AND PARTITIONED DATA SETS TO BE UPDATED

by

I.L. JOHNSTONE

ABSTRACT

The implementation and use of an editor developed for the IBM360 Operating System is described. The editor provides the capability to update both sequential and partitioned data sets as well as providing facilities for printing or punching all or parts of data sets. A feature of this editor is the ability to use many data sets for input or output during update operations. The editor is more versatile and less complicated to use than the editors supplied with the Operating System.

National Library of Australia card number and ISBN 0 642 99586 9

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; DATA PROCESSING; IBM COMPUTERS

CONTENTS

1. INTRODUCTION

This report describes the implementation and usage of an editor developed for use with the IBM System/360 Operating System at the AAEC Research Establishment. It was developed in response to the demand for a comprehensive editor that was both versatile and uncomplicated to use.

IBM System/360 assembler language was the vehicle for implementation of the editor. All input and output is done using the standard BSAM (Basic Sequential Access Method) and BPAM (Basic Partitioned Access Method) Assembler macros.

Standard data sets are associated with each invocation of the editor. These are the command, print (usually the printer) and punch (usually the punch) data sets. The command data set contains the commands for controlling the edit operations, and the data cards to be written to the output(s). The entire command data set is written to the print data set. The records written to the print and punch data sets are controlled by the LIST, PRINT and PUNCH commands.

The editor provides the capability to update both sequential and partitioned data sets with fixed length records (record length not exceeding two hundred and fifty six bytes). An unlimited number of data sets may be specified for output from the editor.

2. COMMANDS

The editor is controlled by the command data set. Two types of cards appear in the command data set, namely command and data cards. Command cards are identified by command character(s) at the start of the card. The default command character is '#' (see Section 2.1.2 for further details and information on how they may be changed). Each command card has a single command on it.

The commands may appear in free format on the card from the column following the command character(s) up to and including column seventy two. Blanks and commas may be inserted where desired between words and delimiters to improve readability. Only the first eight characters of words are significant to the editor.

$$\begin{aligned}
\text{WORD} &= \{\ell a^n \mid \ell \in \text{LETTER and a} \in \text{ALPHA}\ ,\ N \geqslant \emptyset\} \\
\text{LETTER} &= \{A,B,C,\ldots\ldots\ \text{Æ},@,\$,\#\} \\
\text{ALPHA} &= \text{LETTER U } \{\emptyset,\ 1,\ 2,\ \ldots\ldots 8,\ 9\} \\
\text{DELIMITER} &= \{(\ ,\ ),\ -\ \}\ \text{commas are not delimiters.}
\end{aligned}$$

Data cards are those which are not identified as command cards. The sequence number field of the data cards which are not governed by the INSERT

command (see Section 2.2.2) controls the processing of these cards. If a data card has a blank sequence field then the data card is written to the output data set(s) and the input positioning is not affected. If a data card has a sequence number field then the input data set is written to the output data set(s) until a record is found in the input that has a sequence number which equals or excceds the sequence number of the data card. If the sequence numbers are found to be equal then this input record is skipped. The data card is now written to the output data set(s). Attempting to write a data card to an output data set which has a logical record length not equal to eighty will cause an error termination of the editor.

Two command types are recognised, namely active and passive. Active commands cause the operations of the editor to be initiated. Passive commands provide an environment in which edit operations are performed.

2.1 Passive Commands

COLUMN - specify the sequence number field.

USE - specify command character(s).

READ - specify input data set.

WRITE - specify output data set(s).

LIST - specify printing of updated output records.

NUMBER - specify resequencing of output data set(s).

PAGE - skip to new page on print data set.

2.1.1 COLUMN [N [M]]*

N specifies the location of the sequence number field, if omitted column seventy three is assumed. M is the number of digits in the sequence field, the default being 8, and the maximum is 16.

2.1.2 USE CCC...C

The function of this command is to change the command characters which are formed from a variable number of non-blank characters. The new command characters are written as they are to appear on command cards. If no command characters are indicated the '#' is assumed.

Examples

USE ./

USE $

2.1.3 READ ddname [(membername)]

This command specifies the new input data set (see section 3).

---

* [ ] indicates optional specification,

N,M represent decimal numbers not exceeding 16 digits.

Examples

READ SYSUT5

READ PDS (B)

2.1.4 WRITE ddname [(membername)] [ddname [(membername)]] [...]

This command specifies the new output data set(s). Since this command may specify many output data sets any conflicts (see Section3) between these data sets will result in the earlier named data set being closed.

Examples:

WRITE PDS (XY)

WRITE DD1, DD2, DD3 - This example demonstrates the use of optional commas

WRITE PDS(A) PDS(B) - this example demonstrates a simple conflict of data set specification (see Section 3).

The above command is equivalent to WRITE PDS(B).

2.1.5 LIST ; NOLIST ; NO LIST

These commands control the listing of updated records that are being written to the output data set(s).

2.1.6 NUMBER [[FROM] N] [[BY] M]

This command specifies that records written to the output data set(s) will be resequenced. N is the initial value of the sequence numbers supplied, the default being 1ØØ. M is the sequence number increment, the default being 1ØØ.

Specification of a new output data set has no effect on the sequence numbers which are supplied when resequencing is specified.

Examples:

NUMBER 5Ø 5Ø

NUMBER FROM 1ØØØ BY 1ØØØ

NUMBER BY 5Ø

2.1.7 NONUMBER ; NO NUMBER

This command specifies that all records are written to the output data set(s) unchanged.

2.1.8 PAGE

A new page is taken on the print data set.

2.1.9 DEFAULT PASSIVE COMMANDS

```
# COLUMN 73 8
# NO NUMBER
# NOLIST
# READ SYSUT1
```

```
# WRITE SYSUT2
```

2.2 Active Commands

COPY - write input data set to output data set(s).
INSERT - updates from command data set.
SKIP - specify input positioning.
DELETE - specifies deletion of members or records.
PRINT - specifies members or records to be printed.
PUNCH - specifies members or records to be punched.
REWIND - reposition data set to initial record.

Active commands may specify three methods of control namely by sequence number comparisons, card comparisons and by specification of a number of records.

When sequence numbers are being used to control an edit operation, the operation will be terminated when a record with a sequence number equal to or exceeding the specified sequence number is encountered in the input data set. In general no check is made to ensure that all sequence numbers are in ascending order for any input data set. Records with blank sequence fields may appear in any input data set. For these records no comparisons are made, and the operation will not be terminated by such a record.

Commands that are normally controlled by sequence number comparisons may also be controlled by card comparisons; this option is indicated by the absence of sequence number specification from the command. The card following the command card is the basis for the comparison, which proceeds from column one through to the column preceding the sequence number field. The operation continues until a matching input record is found.

2.2.1 COPY

This command specifies the writing of records from the input data set to the output data set(s).

Formats:

1. COPY - the remainder or all of the input is copied.
2. COPY M* [RECORDS]
3A. COPY TO [BEFORE] N**
3B. BEFORE N - alternative to 3A
4A. COPY TO AFTER N
4B. AFTER N - alternative to 4A

---

* a decimal number not exceeding 16 digits

** a decimal number not exceeding 16 digits specifying a sequence number.

2.2.2 INSERT

All data cards following this command are written to the output data set(s) independent of their sequence numbering until the next command card or end-of-file is encountered on the command data set.

2.2.3 SKIP

This command is used to position the input data set.

Formats:

1. SKIP - skip remainder of input.
2. SKIP M [RECORDS]
3. SKIP TO [BEFORE] N
4. SKIP TO AFTER N

2.2.4 DELETE

This command copies selected records from the input data set to the output data set(s) with the deletion of specific records. It can also be used to delete members from partitioned data sets.

Formats:

1. DELETE ddname (membername [membername...]) [.....]
   the specified members are deleted from the indicated partitioned data sets.
2. DELETE M_1 - M_2 M_3 [.....] this format implies:
   COPY TO BEFORE M_1
   SKIP TO AFTER M_2
   COPY TO BEFORE M_3
   SKIP TO AFTER M_3
3. DELETE If delete has no operands then the input data set is copied to the output data set(s) until a card in the input matches the data card following this command. The matching input record is then skipped and the next command processed.

2.2.5 PRINT

This command causes the printing of specified records from the input data set which is positioned to the first record following those printed. The absence of the print data set causes skipping of specified records.

Formats:

1. PRINT - print the remainder or all of input data set
2. PRINT M [RECORDS]

3. PRINT TO [BEFORE] N

4. PRINT TO AFTER N

2.2.6 PUNCH

This command causes the punching of specified records from the input data set which is positioned to the first record following those punched. The absence of the punch data set causes skipping of specified records.

Formats:

1. PUNCH - punch the remainder or all of input data set
2. PUNCH M [RECORDS]
3. PUNCH TO [BEFORE] N
4. PUNCH TO AFTER N

2.3 The Command Data Set

When the end-of-file on the command data set is reached the remainder of the current input is written to the output data set(s). The absence of the command data set specifies that SYSUT1 (default input data set) is copied to SYSUT2 (default output data set).

3. SPECIFICATION OF DATA SETS

The default input and output data sets are assumed to be specified on the SYSUT1 DD and SYSUT2 DD cards respectively. If other input or output data sets are required READ and WRITE commands are essential. These commands are checked for logical conflicts and multiple outputs to the same data set result in earlier output requests being terminated.

4. DD STATEMENTS

4.1 SYSPRINT

This DD statement defines the print data set (normally the printer). The logical record length and block size are fixed at one hundred and twenty one bytes with a record format of FBA.

4.2 SYSIN

This DD statement defines the command data set. The logical record length and blocksize are fixed at eighty bytes with a second format of FB.

4.3 SYSPUNCH

This DD statement defines the punch data set (normally the punch). The logical record length and blocksize are fixed at eighty bytes with a record format of FB.

4.4 INPUT / OUTPUT DD STATEMENTS

The editor requires DD statements to define the data sets referred to by the READ, WRITE, REWIND and DELETE commands.

The maximum record length supported is two hundred and fifty six bytes.

If any data set does not have a logical record length and/or blocksize specification then defaults of eighty bytes are used. If the number of buffers (via the BUFNO parameter) is not specified then 2 buffers will be used.

Multivolume and concatenated data sets are not supported.

When an error occurs involving a particular data set then the diagnostic given refers to the data set by ddname and member (if it is a partitioned data set).

5. CONCLUSIONS

The AEUPDATE program provides editing facilities that are flexible and easy to use. The command cards are simpler to use than those of current editors and comprehensive diagnostic error messages are given; this includes the scanning of the remainder of the command data set for further errors after an initial error is detected.

6. ACKNOWLEDGEMENTS

The author thanks Dr. D.J. Richardson for suggesting this problem and for invaluable discussions during the implementation stages. Thanks are also extended to I.J. Hayes, R.P. Backstrom and P.L. Sanger for helpful suggestions.

APPENDIX

EXAMPLE 1

In this example a new member (AA) is to be added to a partitioned data set, with the new member contained in the command data set.

```
// EXEC    PGM=AEUPDATE
//STEPLIB  DD  DSN=AAELIB.LMODA,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  definition of the partitioned data set
//SYSIN    DD  *
#  WRITE   SYSUT2  (AA)
#  INSERT

                (cards that are to form the member AA)

/*
```

EXAMPLE 2

In this example a member (BB) of a partitioned data set is to be updated with a new membername and the prior member deleted from the partitioned data set. New sequence numbers are to be supplied for the new member (CC).

```
// EXEC  PGM=AEUPDATE
//STEPLIB   DD  DSN=AAELIB.LMODA,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//DATA      DD  DSN=DISPLAY,DISP=OLD
//SYSIN     DD  *
# READ      DATA(BB)
# WRITE     DATA(CC)
# NUMBER FROM 1ØØØ by 1ØØØ
# DELETE  1ØØ - 75Ø  1Ø5Ø  135Ø
       data statement 1              ØØØØ14ØØ
       data statement 2              ØØØØ165Ø
# COPY
# DELETE    DATA(BB)

/*
```

EXAMPLE 3

In this example a three-member partitioned data set (NEWPDS) is to be created. Two members (CC and DD) are copied from an existing data set (OLDPDS). A new member (EE) is contained in the command data set.

```
// EXEC   PGM=AEUPDATE
//STEPLIB   DD  DSN=AAELIB.LMODA,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//IN        DD  DSN=OLDPDS,DISP=SHR
//OUT       DD  DSN=NEWPDS,DISP=(NEW,KEEP),UNIT=SYSDA,
//              SPACE=(CYL,(1,2,4)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSIN     DD  *
# WRITE OUT (CC)
# READ IN   (CC)
# COPY
# WRITE OUT (DD)
# READ  IN  (DD)
# COPY
# WRITE OUT (EE)
# INSERT
              (cards that are to be the member EE)
```

PRINTED OFFSET BY THE
AAEC RESEARCH ESTABLISHMENT