CMS Pipelines Visualized

March 11, 1994

Yuichi Ishikawa
IBM Japan

VNET : ISHIKAWA at TOKVMTR1
Internet : yishikawa@vnet.ibm.com

Notice

This book was made with Melinda Varian's and John Hartmann's encouragement. Melinda Varian, of Princeton University, reviewed this document and made intensive editing and many helpful suggestions. Without their help, this document could not be made.

The pictorial diagrams in this book are translated from a Japanese document "ZUSETSU *CMS Pipelines* Guide Book" (GE88-0024), written by Yuichi Ishikawa. The description of each program is from John P. Hartmann's description and edited partially for this document.

Contents

1.0 Introduction

This document is intended to help the user of *CMS Pipelines* understand the function of programs with pictorial diagrams. *CMS Pipelines* has rich programs, and the pictorial functional diagrams will help the user grasp the functions of many programs intuitively and quickly. Users can SEE the functions rather than READing descriptions. It will also help the user in selecting programs to perform the desired tasks.

Note that this document describes only the major programs that are frequently used, and major functions of each of them.

Level of *CMS Pipelines*

This book is applicable to VM/ESA R2.1 or higher. Some of the programs described in this document are not available on VM/ESA R1.1, R2.0, 5758-RAC *CMS Pipelines* Program Offering, and 5799-DKF *CMS Pipelines* PRPQ.

IBM documents

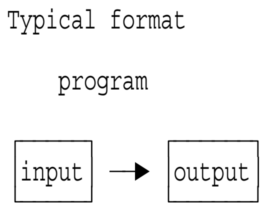
You should obtain a copy of VM/ESA *CMS Pipelines* Reference (English edition SC244-5592, Japanese edition SC88-6098) for the complete and detailed description of programs. Japanese users are recommended to obtain a copy of "ZUSETSU *CMS Pipelines* guide book" (GE88-0024); it is written in Japanese.

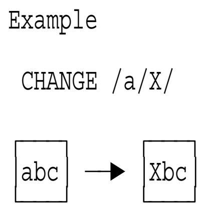
Notation of pictorial diagrams

The pictorial diagrams are described in the following format.

Typical format

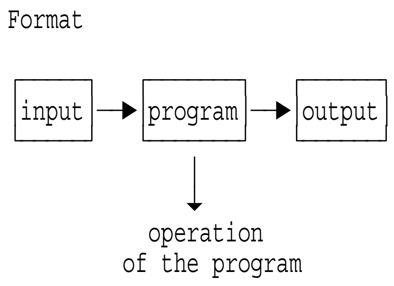
Typically, the left box shows input records, and it is transformed into the right box which is output records.

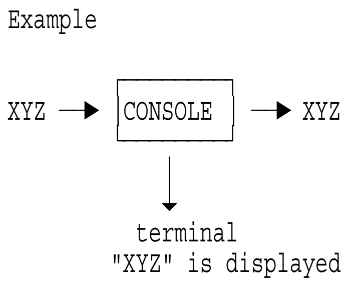




Other format

Some programs are drawn differently as appropriate to describe their function.

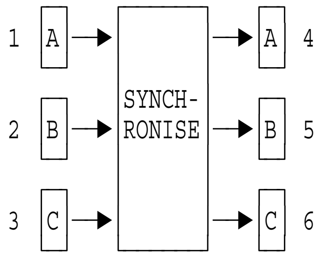




Multiple streams

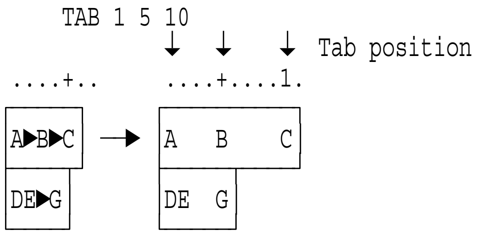
If a stage has multiple streams connected, the upper left arrow is the primary stream, the next lower arrow is the secondary stream, and so on.

Digits may be written beside input or output records. They are the sequence number of the operation. In the following example, 1) record “A” is read, 2) record “B” is read, 3) record “C” is read, and then 4) record “A” is written, and so on.



Scale There may be a scale and other additional pointers to make column-dependent operations clearer.

Assume ▶ is a tab character

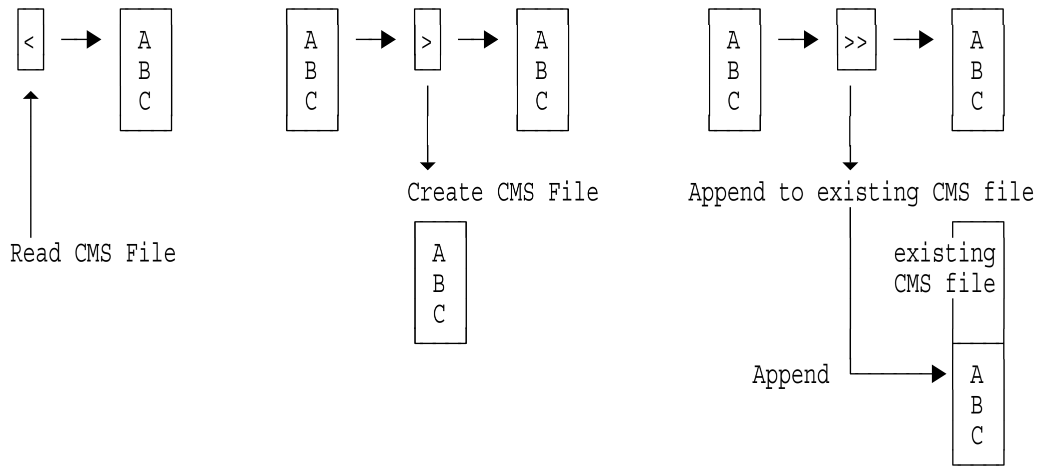


2.0 Frequently used device drivers

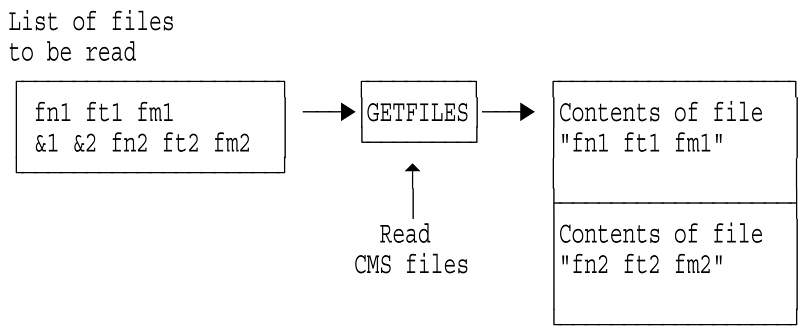
< reads a CMS file from storage, from a minidisk, or from a Shared File System (SFS) directory that has been accessed with a mode letter. The file must exist.

> replaces a file on a minidisk or in a Shared File System (SFS) directory that has been accessed with a mode letter. A file is created if one does not exist.

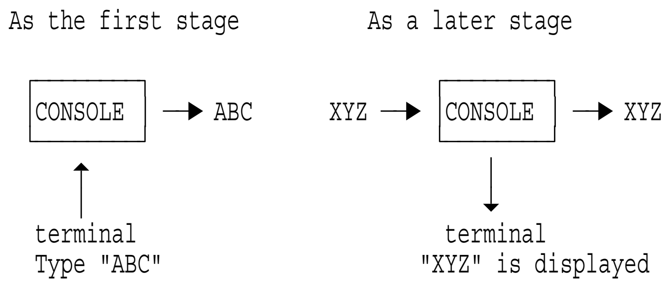
> > appends records to a file on a minidisk or in a Shared File System (SFS) directory that has been accessed with a mode letter. A file is created if one does not exist.



GETfiles reads the contents of files into the pipeline. The files to read (as defined for <) are specified in the input records.

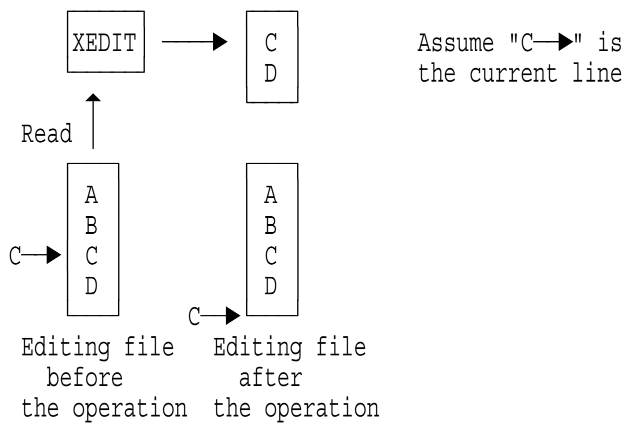


CONSOLE connects the virtual machine console to the pipeline. When CONSOLE is first in a pipeline, it reads lines from the terminal and writes them into the pipeline. When CONSOLE is not first in a pipeline, it copies lines from the pipeline to the terminal.

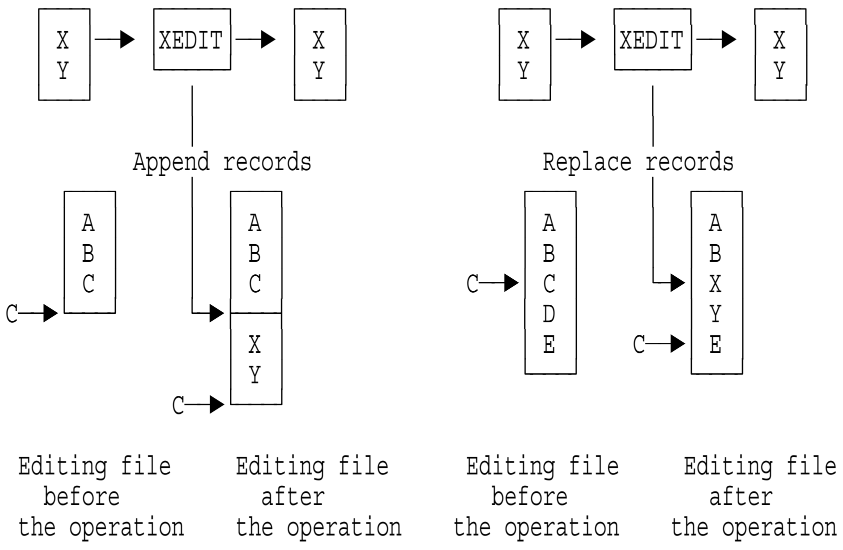


XEDIT connects the pipeline to a file in the XEDIT ring; it reads lines from the file into the pipeline or writes lines from the pipeline into the file.

As the first stage


XEDIT
C
D
Assume "C→" is the current line
Read
A
B
C→ C
D
A
B
C
C→ D
Editing file before the operation
Editing file after the operation

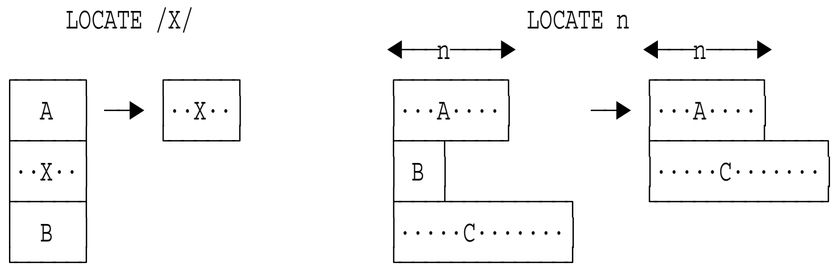
As a later stage As a later stage


X
Y
XEDIT
X
Y
Append records
A
B
C
C→
A
B
C
X
Y
C→
Editing file before the operation
Editing file after the operation
X
Y
XEDIT
X
Y
Replace records
A
B
C→ C
D
E
A
B
X
Y
C→ E
Editing file before the operation
Editing file after the operation

3.0 Selecting records

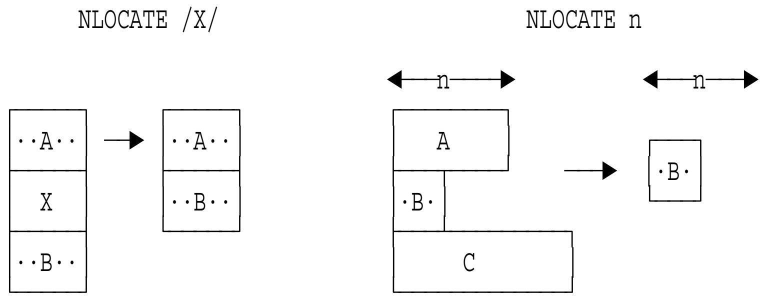
LOCATE /X/ selects records that contain string “X”.

LOCATE n selects records that are at least as long as “n”. (n is positive number)



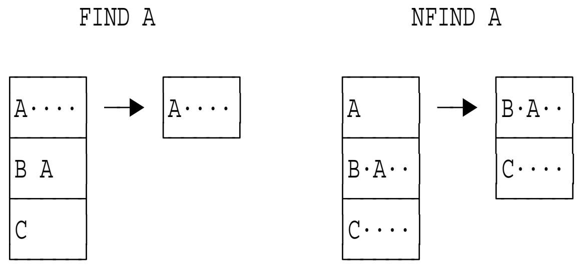
NLOCATE /X/ selects records that do not contain string “X”.

NLOCATE n selects records that are shorter than “n”. (n is positive number)

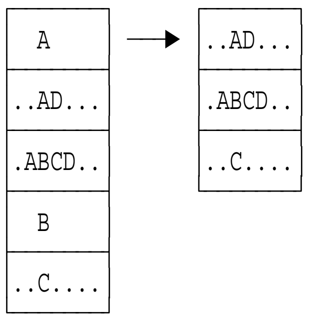


FIND selects records that begin with the specified string.

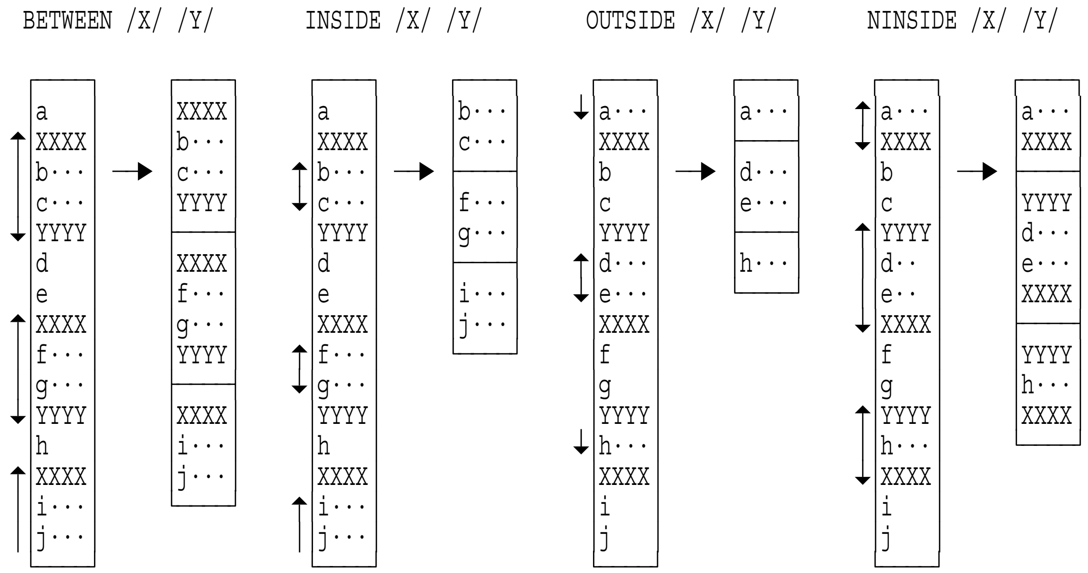
NFIND selects records that do not begin with the specified string.



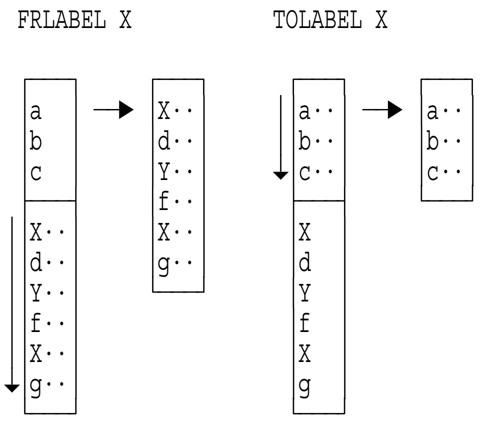
ALL (/A/ & /D/) ! /C/
selects lines that contains “A” and “D”, or lines that contains “C”. ALL selects records that satisfy a specified search criterion.



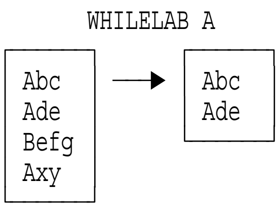
BETWEEN /X/ /Y/	selects groups of records whose first record begins with “X”. The record at the end of each group begins with “Y”.
INSIDE /X/ /Y/	selects groups of records whose first record follows a record that begins with “X”. The last record of each group is followed by a record that begins with “Y”.
OUTSIDE /X/ /Y/	discards groups of records whose first record begins with “X” and whose last record begins with “Y”.
NINSIDE /X/ /Y/	discards groups of records whose first record follows a record that begins with “X” and whose last record precedes a record that begins with “Y”.


BETWEEN /X/ /Y/
INSIDE /X/ /Y/
OUTSIDE /X/ /Y/
NINSIDE /X/ /Y/
a XXXX b··· c··· YYYY d e XXXX f··· g··· YYYY h XXXX i··· j···
XXXX b··· c··· YYYY XXXX f··· g··· YYYY XXXX i··· j···
b··· c··· f··· g··· i··· j···
a··· XXXX b c YYYY d··· e··· XXXX f g YYYY h··· XXXX i j
a··· d··· e··· h···
a··· XXXX b c YYYY d·· e·· XXXX f g YYYY h··· XXXX i j
a··· XXXX YYYY d··· e··· XXXX YYYY h··· XXXX

FRLABEL X	discards input records up to the first one that begins with the specified string “X”. That record and the records that follow are selected.
TOLABEL X	selects input records up to the first one that begins with the specified string “X”. That record and the records that follow are discarded.

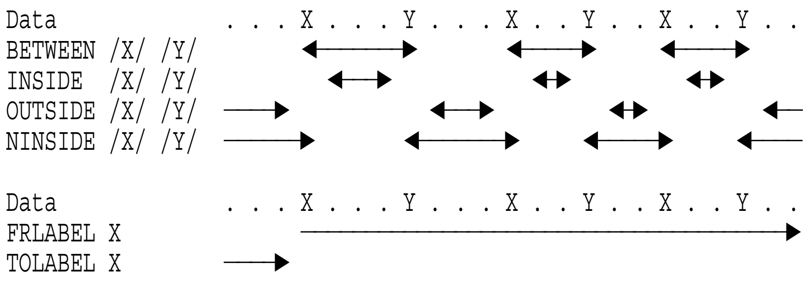


WHILELAB selects input records up to the first one that does not begin with the specified string. That record and the records that follow are discarded.



Summary of BETWEEN, INSIDE, OUTSIDE, NINSIDE, FRLABEL and TOLABEL

Assume that records run from left to right. “X”, “Y”, and a period “.” in data represent records. An arrows indicates selected records.

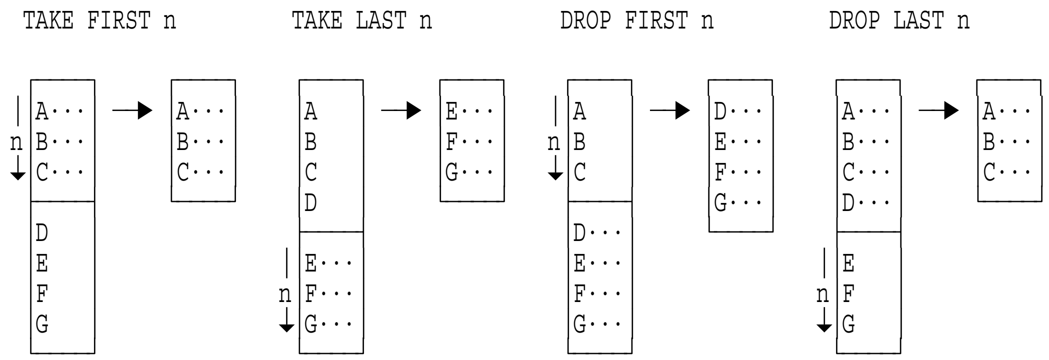

Data . . . X . . . Y . . . X . . Y . . X . . Y . .
BETWEEN /X/ /Y/
INSIDE /X/ /Y/
OUTSIDE /X/ /Y/
NINSIDE /X/ /Y/
Data . . . X . . . Y . . . X . . Y . . X . . Y . .
FRLABEL X
TOLABEL X

TAKE FIRST n selects the first n records and discards the remainder.

TAKE LAST n discards records up to the last n and selects the last n records.

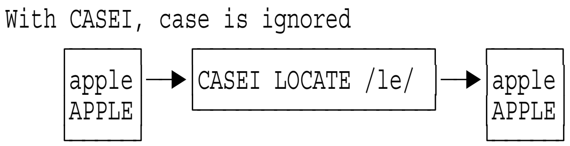
DROP FIRST n discards the first n records and selects the remainder.

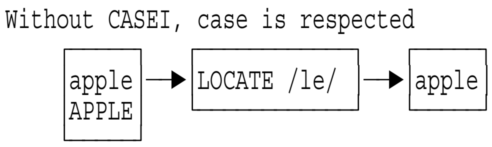
DROP LAST n selects records up to the last n and discards the last n records.



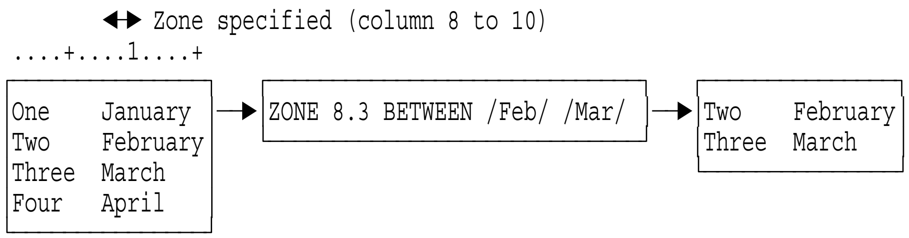
3.1 Augmenting selection stages

CASEI runs a selection stage ignoring case.

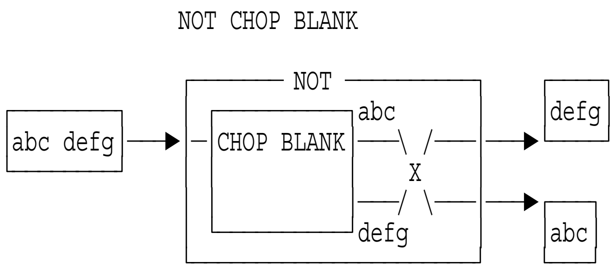




ZONE runs a selection stage against a specified column range in the input records.



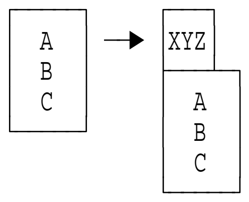
NOT runs a stage inverting the primary output stream and the secondary output stream.



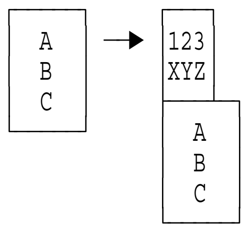
4.0 Inserting records

LITERAL writes its argument string into the pipeline and then passes records on the input to the output stream. If LITERAL is the first stage, it just writes a record with the specified string. Note that if more than one LITERAL is specified, the record from the last one is first in the pipeline.

```
LITERAL XYZ
```



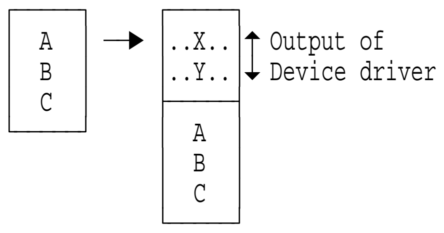
```
LITERAL XYZ| LITERAL 123
```



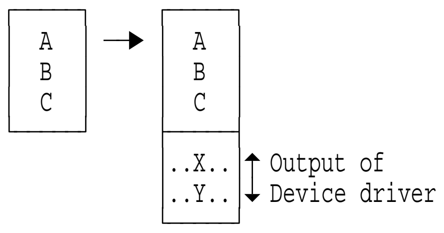
PREFACE runs a device driver to generate output which is passed to PREFACE’s output. It then passes all its input records to the output.

APPEND passes all input records to the output and then runs a device driver to generate additional output.

```
PREFACE device-driver
```



```
APPEND device-driver
```



5.0 Rearranging records

SPECs input output (input output) (input output) .....

builds output records from the contents of input records and literal fields. It can convert the contents of fields in several ways. It can generate an output record containing data from several input records, and it can generate several output records from a single record.

SPECs colrange colrange

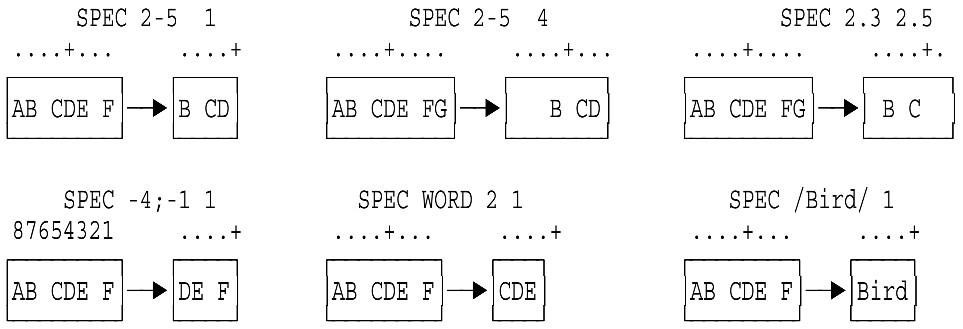
reads characters in specified columns and places them at the specified position in the output record. The input column numbers may be specified as negative numbers, in which case they are counted from the ends of the records. A range of negative numbers is indicated by placing a semi-colon between them; thus, “-5;-3” indicates the 5th-from-last through 3rd-from-last columns.

SPECs WORD n colrange

reads the N-th word and places it at the specified position.

SPECs /string/ colrange

writes the specified string to the specified position.

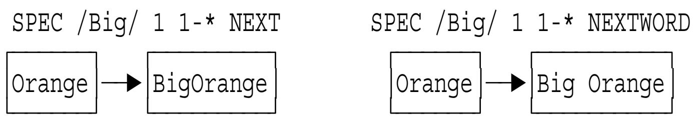


SPECs input output input NEXT

The output position NEXT indicates that the data are put immediately after the right-most item that has been put in the output record so far.

SPECs input output input NEXTWORD

NEXTWORD is similar to NEXT, but NEXTWORD appends a blank to the output record before appending data.



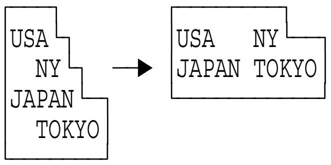
SPECs input output READ input output

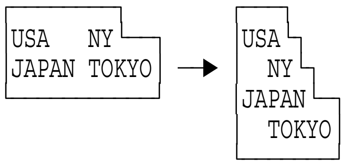
READ releases a record and locates the next record. You can write input fields from consecutive input records to an output record.

SPECs input output WRITE input output

WRITE writes the output record built so far, leaving current output record empty. You can write multiple output records based on the contents of an input record.

```
  SPEC 1.5 1 READ WORD 1 7          SPEC WORD 1 1 WRITE WORD 2 3
 ....+..      ....+....1...       ....+....1...     ....+..
```

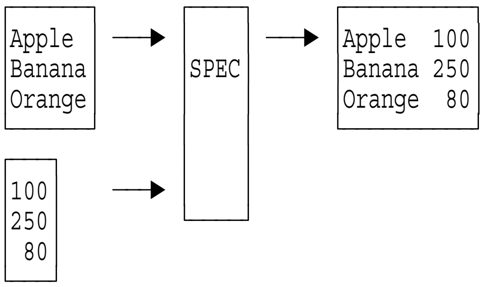




SPECs input output SELECT stream input output

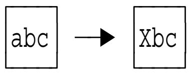
SELECT specifies that subsequent input fields refer to the specified input stream, which is specified by its stream number. (Streams are numbered 0, 1, 2, etc.)

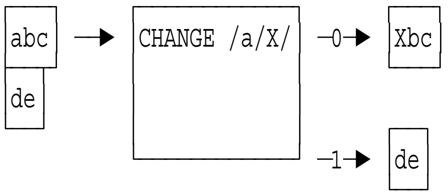
```
  SPEC 1-* 1  SELECT 1  1-* 8
                       ....+....1
```



CHANGE /a/X/ replaces occurrences of string “a” with string “X”. If the secondary output is defined, changed records are written to the primary output, and unchanged records are passed to the secondary output.

```
CHANGE /a/X/              If secondary output is defined
```



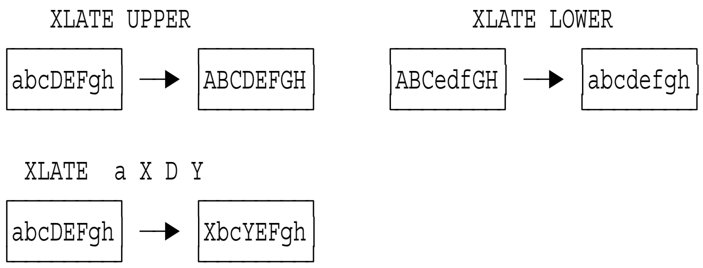


XLATE transliterates the contents of records in accordance with a translate table.

XLATE UPPER uppercases lower-case letters.

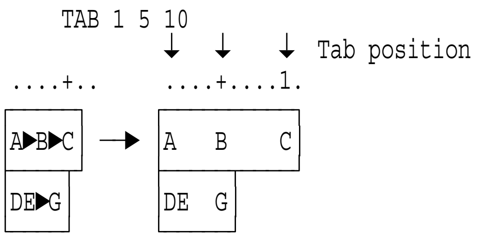
XLATE LOWER lowercases upper-case letters.

XLATE a X D Y translates “a” to “X” and “D” to “Y”.



UNTAB expands tab characters (X'05') in the record to blanks to line up columns.

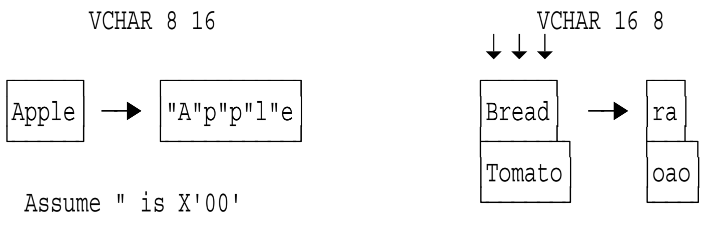
Assume ▶ is a tab character



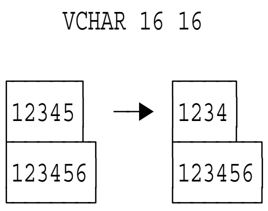
VCHAR changes the character length, inserting or discarding high-order bits. Input and output records are considered to contain characters of the length specified, spanned over bytes.

Insert X'00' before each character

Remove every other character

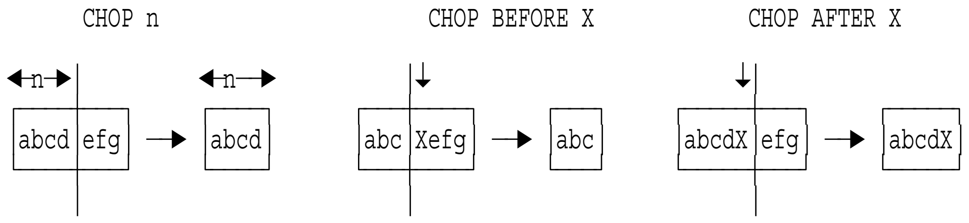


Make the record length even

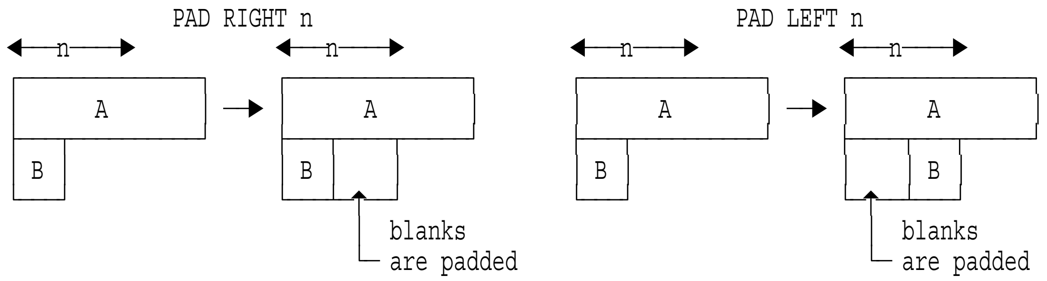


6.0 Cutting and Pasting

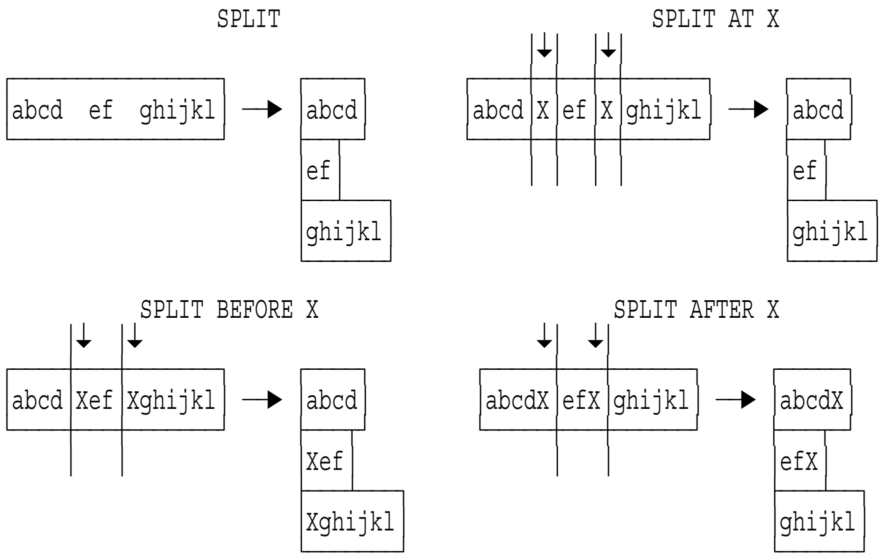
CHOP n	truncates records after specified column "n".
CHOP BEFORE c	truncates records before specified character or string.
CHOP AFTER c	truncates records after specified character or string.



PAD RIGHT n	expands records that are shorter than "n". Specified pad character (default is blank) is added to the end of the record.
PAD LEFT n	inserts pad blanks at the beginning of the record.



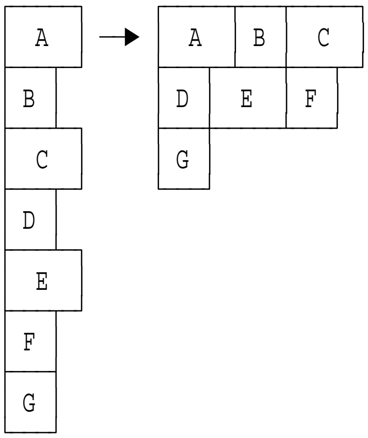
SPLIT splits records and writes one or more records based on the contents of an input record; each part ends before or after a specified character or string. (The default is to split at blanks.)



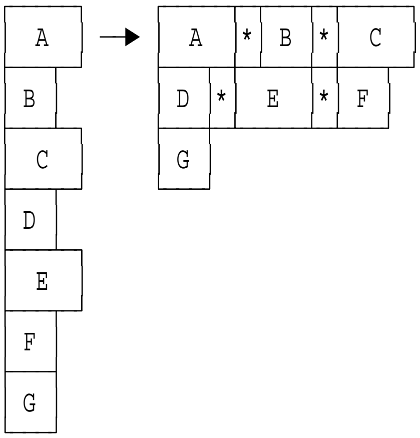
JOIN n puts n + 1 input records together into one output record.

JOIN n /*/ puts n + 1 input records together into one output record with the specified string "*" between them.

JOIN 2

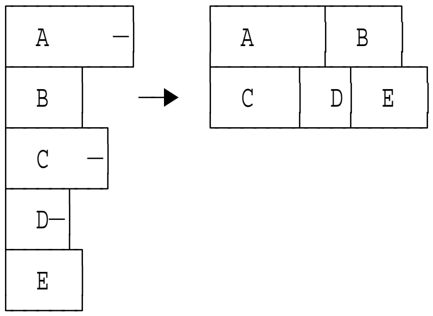


JOIN 2 /*/

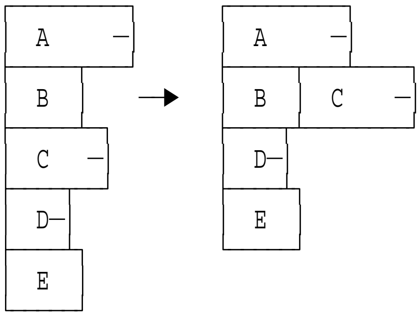


JOINCONT joins records that are marked with a continuation string at the end of the continued record or at the beginning of the continuation record

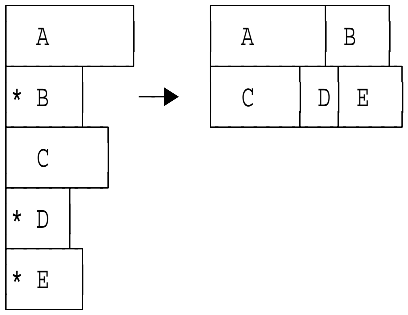
JOINCONT TRAILING /–/



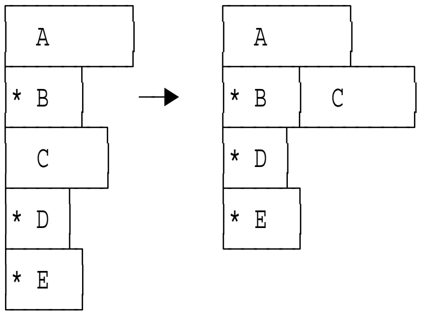
JOINCONT NOT TRAILING /–/



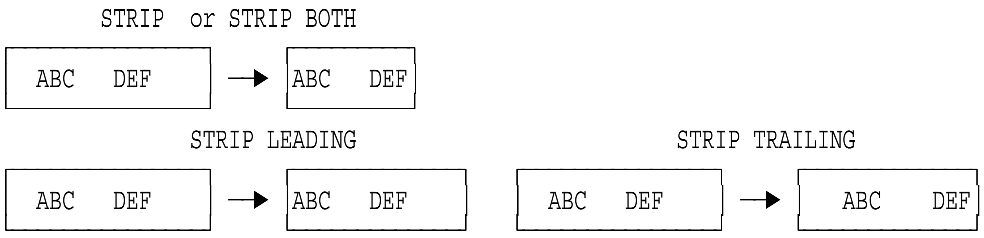
JOINCONT LEADING /*/



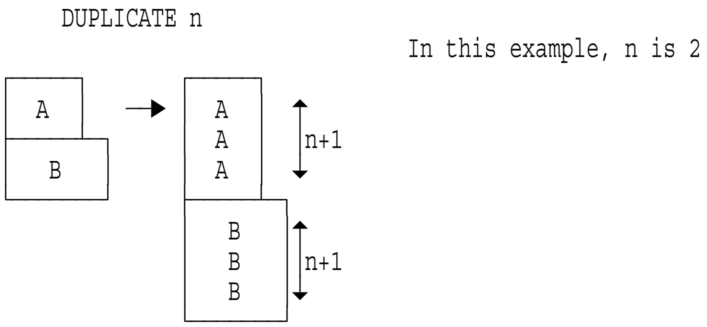
JOINCONT NOT LEADING /*/



STRIP removes specified leading or trailing characters (default is blank) or strings.

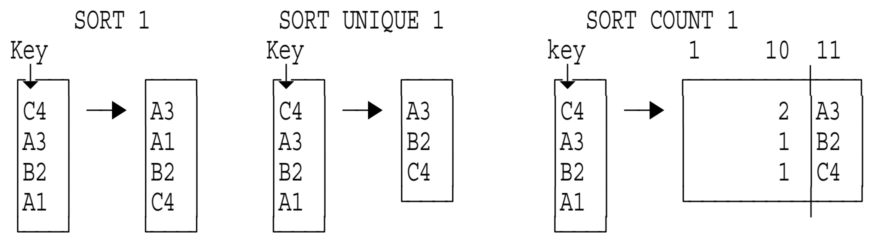


DUPLICATE n writes each input record into the pipeline n + 1 times.

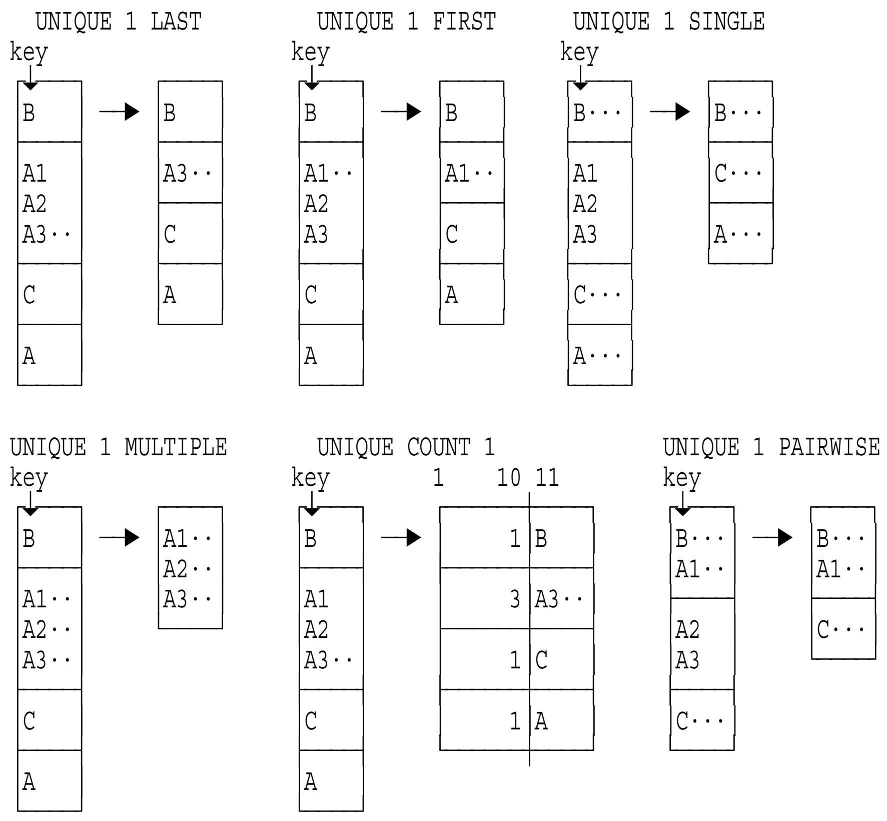


7.0 Sorting and getting Unique records

SORT	sorts records.
SORT UNIQUE	sorts records, retaining only the first record with a given key. Subsequent records with duplicate keys are discarded.
SORT COUNT	sorts records, retaining only the first record with a given key. Subsequent records with duplicate keys are discarded. A 10-character count of the number of occurrences of the key is prefixed to the output record.

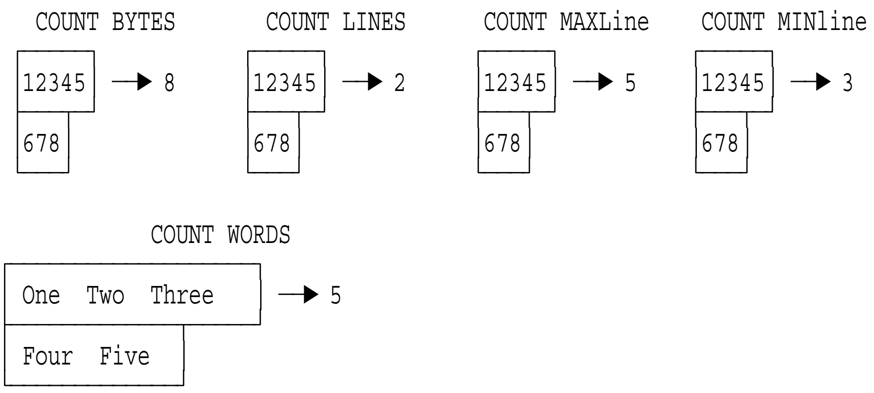


UNIQUE FIRST	selects the first record in a run of records having a particular key, discarding further records with this key.
UNIQUE LAST	selects the last record in a run of records having a particular key, discarding the earlier records with this key.
UNIQUE SINGLE	selects records for which there are no duplicates.
UNIQUE MULTIPLE	selects records for which there are duplicates.
UNIQUE PAIRWISE	selects pairs of records that are not duplicates.

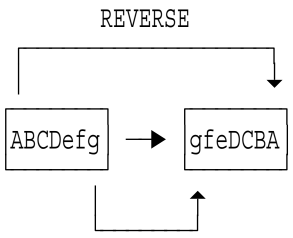


8.0 Other filters

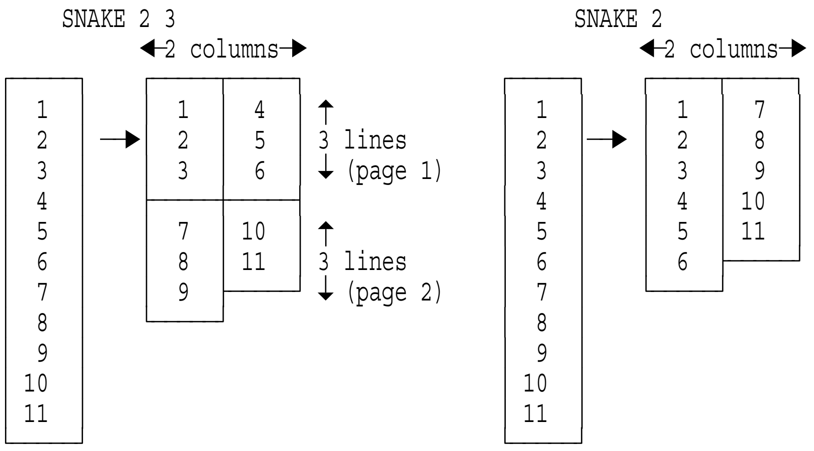
COUNT counts the number of input lines, words, characters, or any combination thereof. It can also report the length of the shortest record, or the longest record, or both. It writes a line with the specified counts at end-of-file.

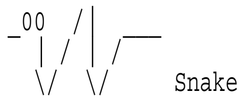


REVERSE reverses the contents of each record so that the first character becomes the last, the last character becomes the first, the penultimate character becomes the second, the second character becomes the penultimate, and so on.

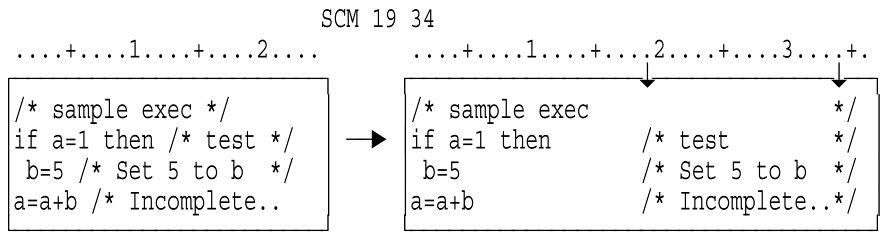


SNAKE breaks the input file into columns of the specified depth and pastes the columns together side by side. Thus, the input file wiggles its way across the page like a snake.





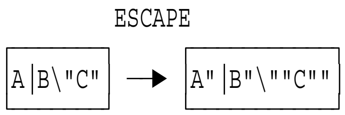
SCM processes REXX and C programs to line up comments and complete unclosed comments.



```
                          SCM 19 34
....+....1....+....2....      ....+....1....+....2....+....3....+.

/* sample exec */             /* sample exec                 */
if a=1 then /* test */    →   if a=1 then       /* test       */
 b=5 /* Set 5 to b  */         b=5              /* Set 5 to b */
a=a+b /* Incomplete..         a=a+b             /* Incomplete..*/
```

ESCAPE processes records to insert escape characters in front of characters that are specified as needing to be escaped. The default characters to be escaped are vertical bar (|), back slash (\), and double quotation ("). The default escape character is double quotation.

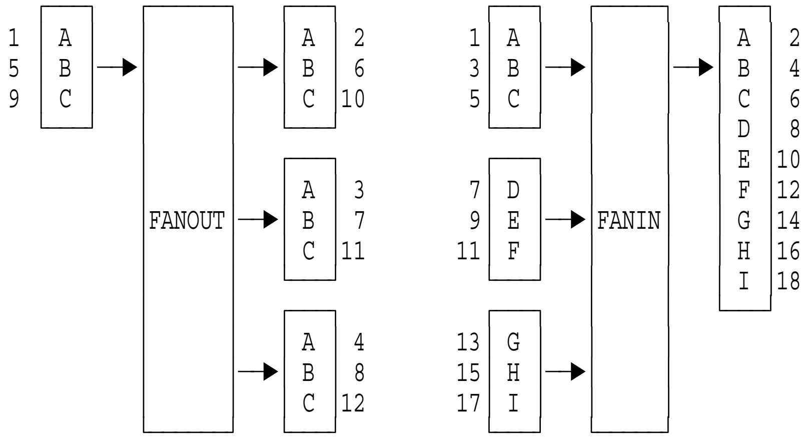


```
          ESCAPE
A|B\"C"  →  A"|B"\""C""
```

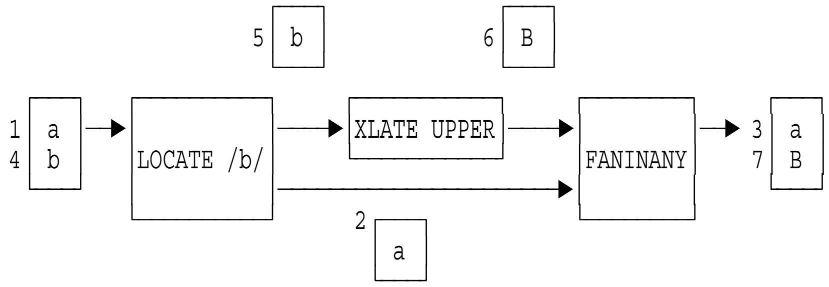
9.0 Merging, joining streams (Gateways)

FANOUT writes a copy of each input record to the primary output stream, the secondary output stream, and so on.

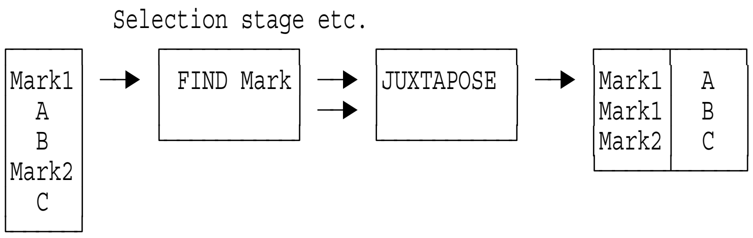
FANIN passes all records on the primary input stream to the primary output stream, then all records on the secondary input stream to the primary output stream and so on.



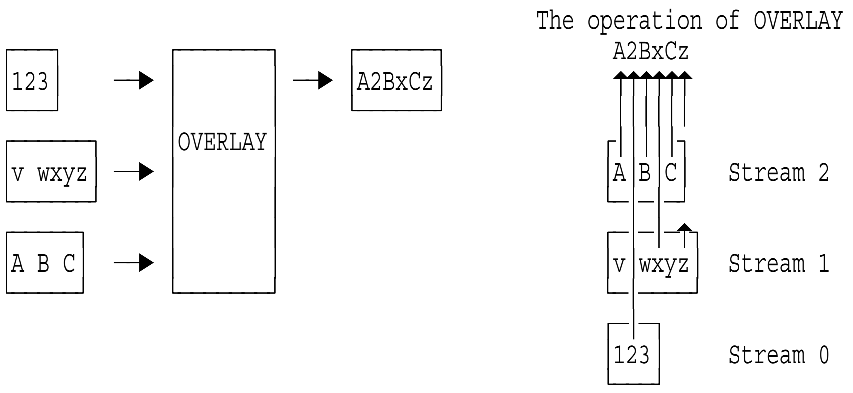
FANINANY copies records from its input streams to the primary output stream. It reads records from whatever input stream has one ready.



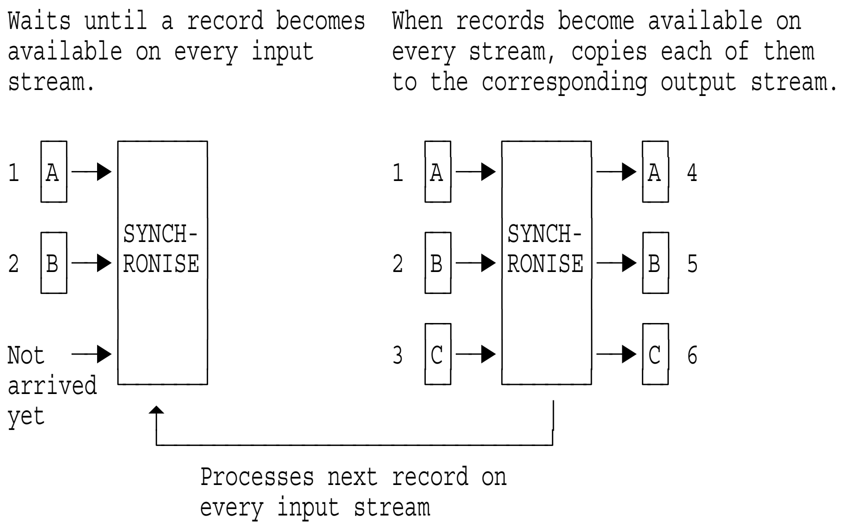
JUXTAPOSE prefixes a record from the primary input stream to the records from the secondary input stream that become available after the record from the primary input stream. When the next record from the primary input stream is read, it becomes the new prefix.



OVERLAY combines a record from each connected input stream into a single output record. Each position in the output record contains the character from the highest numbered input stream where the corresponding position is present and not equal to the specified pad character (default is blank).



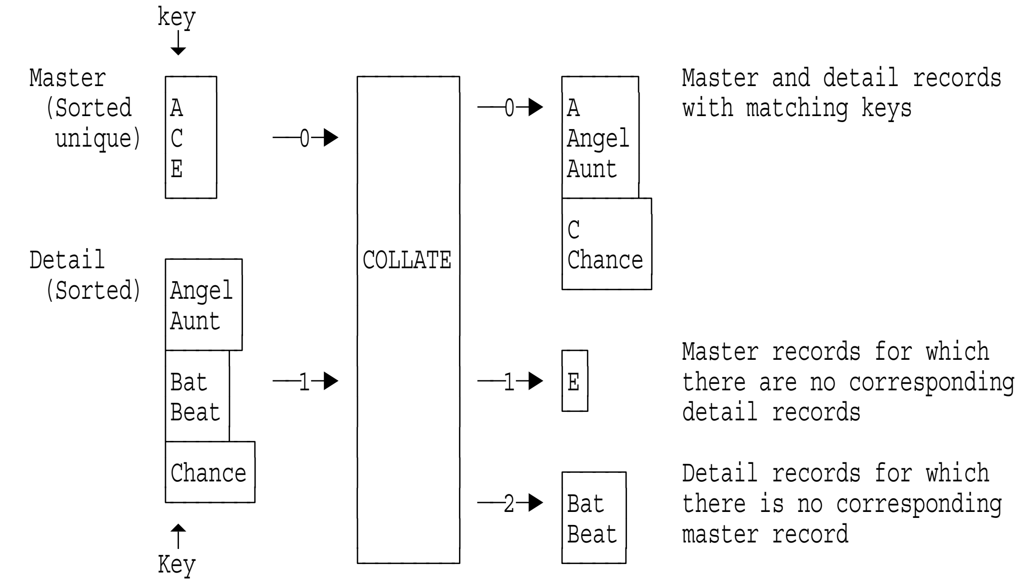
SYNCHRONISE forces records on parallel streams of a pipeline to move in unison through the pipeline. SYNCHRONISE waits until there is a record available on every input stream and then copies one record from each input stream to the corresponding output stream. It copies no further records to its output until there is again a record available on each input stream. With SYNCHRONISE, the records on one stream can be used to pace the flow through the pipeline of the records on some other stream.



Waits until a record becomes available on every input stream.
When records become available on every stream, copies each of them to the corresponding output stream.
1 A
2 B
Not arrived yet
SYNCH-RONISE
1 A
2 B
3 C
SYNCH-RONISE
A 4
B 5
C 6
Processes next record on every input stream

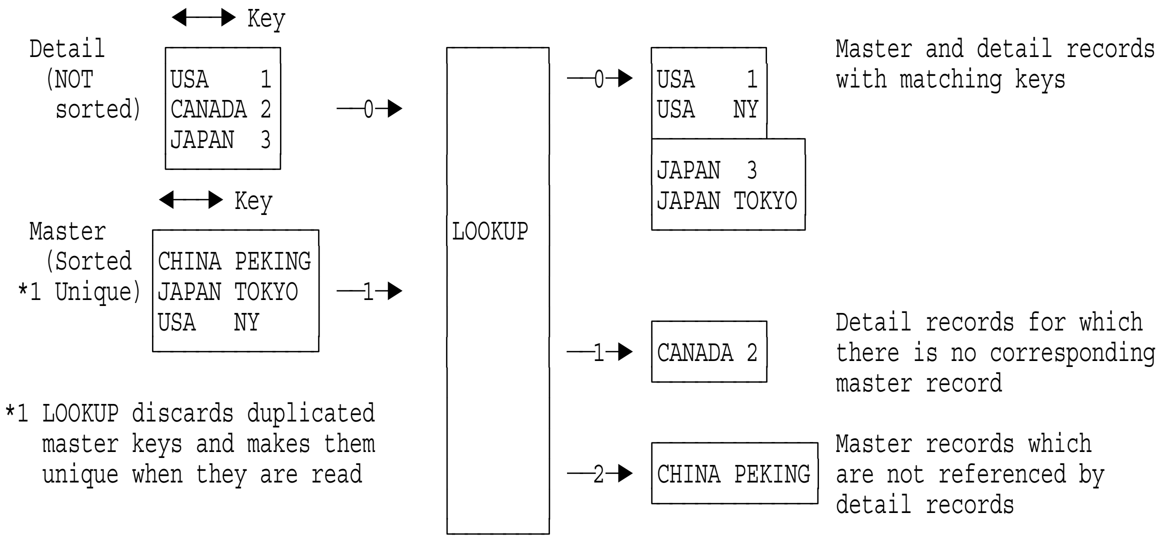
10.0 Matching records

COLLATE compares two input streams containing master records and detail records. Depending on the contents of a key field in the records, input records are passed to one of three output streams (if connected) or discarded.



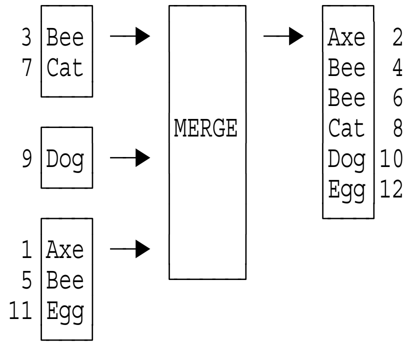
key
Master (Sorted unique)
A
C
E
Detail (Sorted)
Angel
Aunt
Bat
Beat
Chance
Key
0
1
COLLATE
0
A
Angel
Aunt
C
Chance
Master and detail records with matching keys
1
E
Master records for which there are no corresponding detail records
2
Bat
Beat
Detail records for which there is no corresponding master record

LOOKUP processes an input stream (detail records) against a reference (master records), comparing a key field. When a detail record has the same key as a reference record, one or both of the records are passed to the primary output stream; unmatched detail records are passed to the secondary output stream; unmatched reference records are passed to the tertiary output stream when end-of-file is reflected on the primary input stream.



Key
Detail (NOT sorted)
USA 1
CANADA 2
JAPAN 3
0
Key
Master (Sorted *1 Unique)
CHINA PEKING
JAPAN TOKYO
USA NY
1
*1 LOOKUP discards duplicated master keys and makes them unique when they are read
LOOKUP
0
USA 1
USA NY
JAPAN 3
JAPAN TOKYO
Master and detail records with matching keys
1
CANADA 2
Detail records for which there is no corresponding master record
2
CHINA PEKING
Master records which are not referenced by detail records

MERGE merges multiple input streams down to a single output stream, interleaving the records according to the contents of their key fields. The input streams should already be in the specified order; this is not verified.



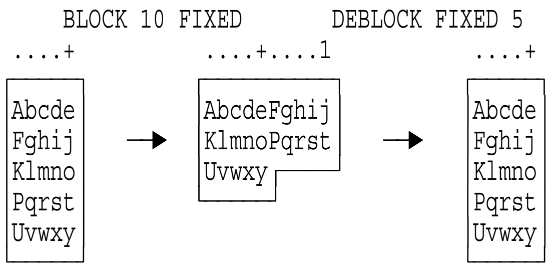
11.0 Converting record formats

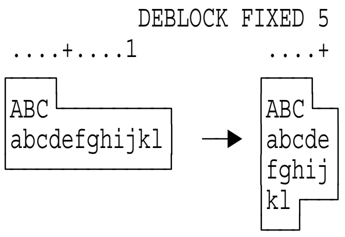
BLOCK generates output blocks from input logical records. The output blocks are in a format that is suitable for interchange with other systems. The format is determined by the operand specified.

DEBLOCK The converse of BLOCK.

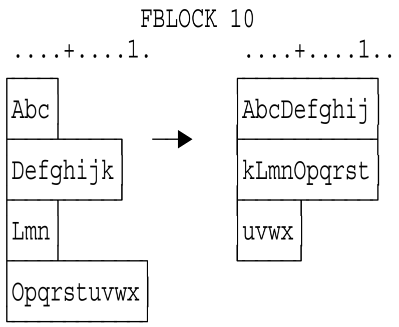
BLOCK n FIXED juxtaposes records with no control information in between. All input record lengths must be the same. The length of the output record is “n” and it is a multiple of the input record length.

DEBLOCK FIXED n splits input records with each n byte length. It is the converse of BLOCK n FIXED.



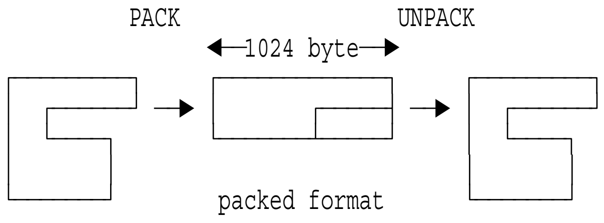


FBLOCK writes fixed-length records containing data from one or more input records. Data from an input record can span output records.



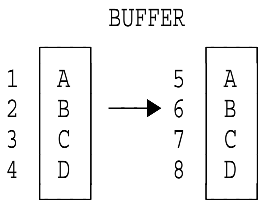
PACK packs the input records into 1024-byte output records in the format used by COPYFILE and XEDIT. The last record is padded with binary zeros.

UNPACK turns a packed file back into plain records. It does not modify a file that is not packed.



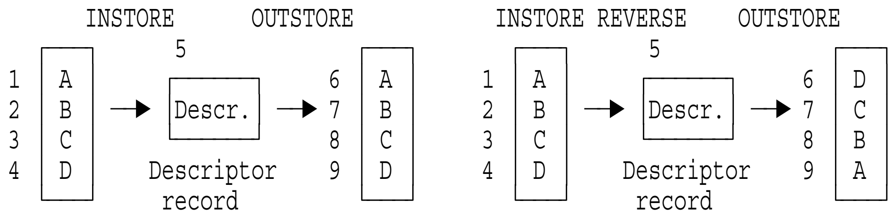
12.0 Buffering records

BUFFER reads input records and accumulates them in memory. BUFFER writes the buffered records to its output when it reaches end-of-file on its input.

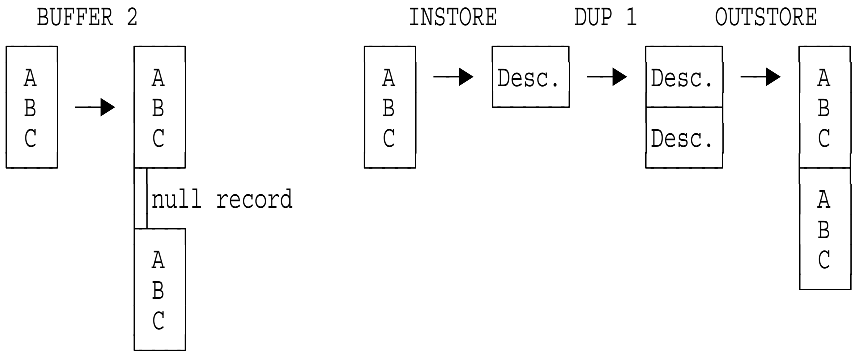


INSTORE stores all input records in a data structure and then writes a single descriptor record into the pipeline. A cascade of INSTORE REVERSE and OUTSTORE reverses the order of records.

OUTSTORE recreates the records from the data structure built by INSTORE.



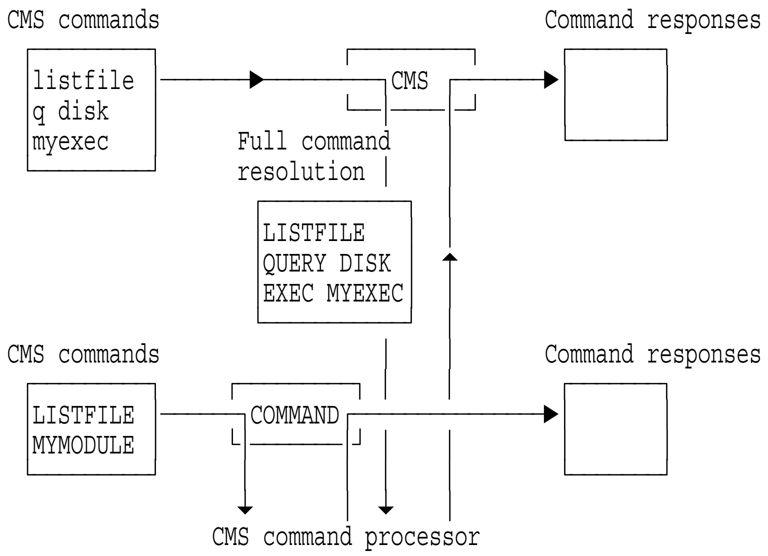
Both BUFFER and INSTORE/OUTSTORE can make copies of all input records. However, BUFFER inserts a record (the default is a null record) between copies.



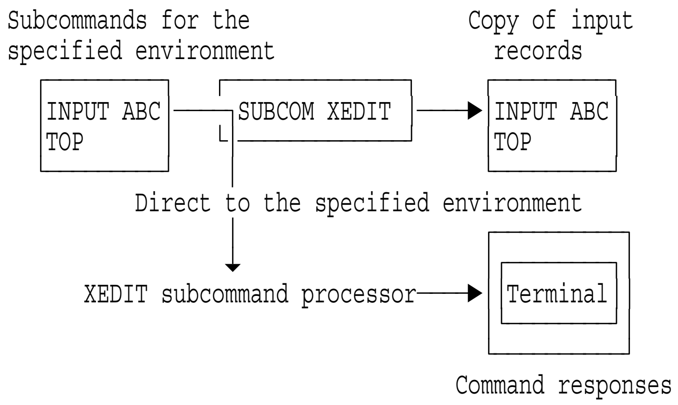
13.0 Host command interfaces

CMS issues CMS commands with full command resolution, and captures the command response, which is then written to the output rather than being displayed on the terminal.

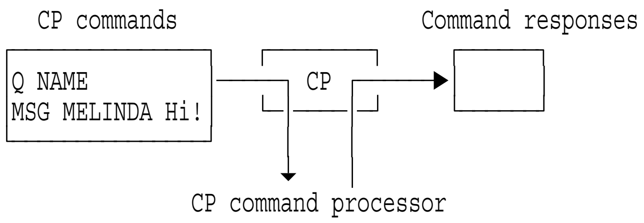
COMMAND issues CMS commands that can be resolved to modules or CMS nucleus routines and captures the command response, which is then written to the output rather than being displayed on the terminal.



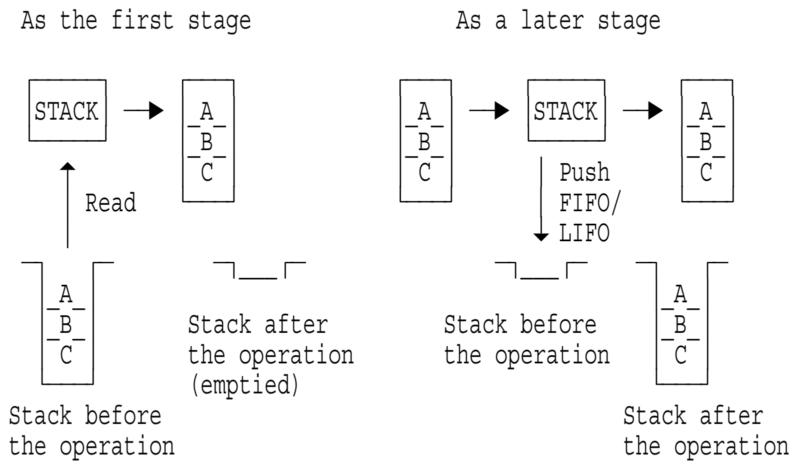
SUBCOM issues commands to a subcommand environment without intercepting terminal output. Command responses are displayed on the terminal.


Subcommands for the specified environment
INPUT ABC
TOP
SUBCOM XEDIT
Copy of input records
INPUT ABC
TOP
Direct to the specified environment
XEDIT subcommand processor
Terminal
Command responses

CP issues CP commands and captures the command response, which is then written to the output rather than being displayed on the terminal.

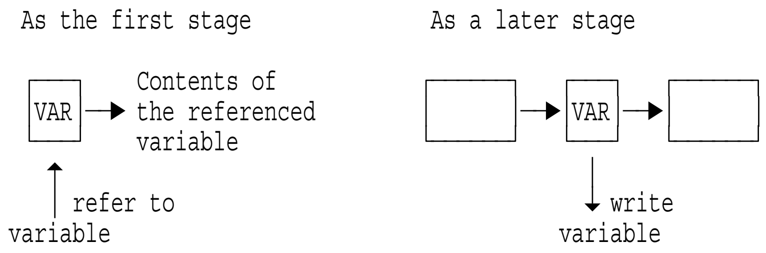


STACK connects the console stack to the pipeline. When STACK is first in a pipeline, it reads lines from the console stack into the pipeline. When STACK is not first in a pipeline, it copies the lines in the pipeline onto the program stack.

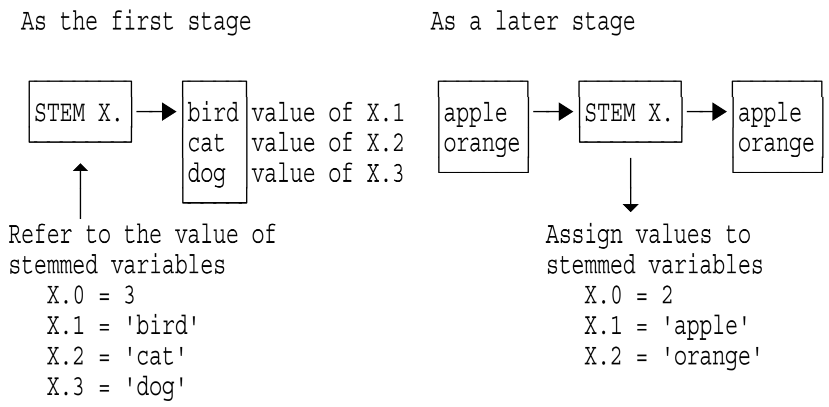

As the first stage
STACK
A
B
C
Read
A
B
C
Stack after the operation (emptied)
Stack before the operation
As a later stage
A
B
C
STACK
A
B
C
Push FIFO/ LIFO
Stack before the operation
A
B
C
Stack after the operation

14.0 Accessing variables

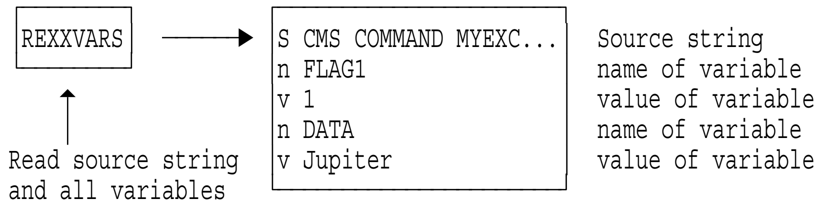
VAR connects a variable to the pipeline. When VAR is first in the pipeline, the contents of the specified variable are written to the pipeline. When VAR is not first in a pipeline, it sets the specified variable to the contents of the first input record and then passes all input to the output; the variable is dropped if there is no input.



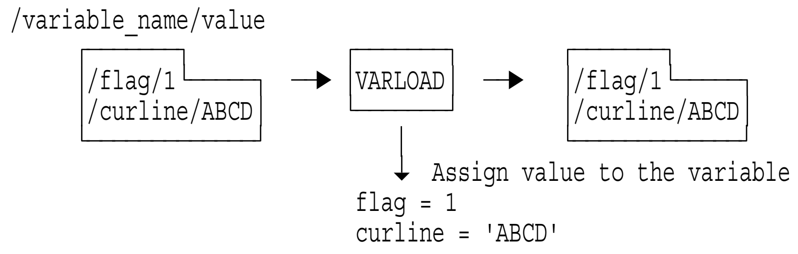
STEM connects a stemmed array of variables to the pipeline. When STEM is first in the pipeline, the contents of the array are written to the pipeline; an array is built when STEM is not first in a pipeline. A stemmed array consists of variables that have names ending in a non-negative integer (the index). The variable with index 0 contains the count of “data” variables, which are numbered from 1 onwards.



REXXVARS writes the names and values of currently exposed REXX variables (including the source string) into the pipeline. REXXVARS can retrieve variables from the current EXEC (or REXX pipeline program) or from one of its ancestors.

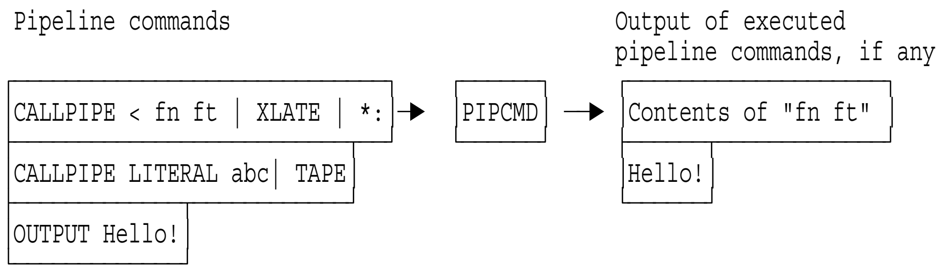


VARLOAD sets the values of variables based on the contents of its input records.



15.0 Executing pipeline or pipeline commands

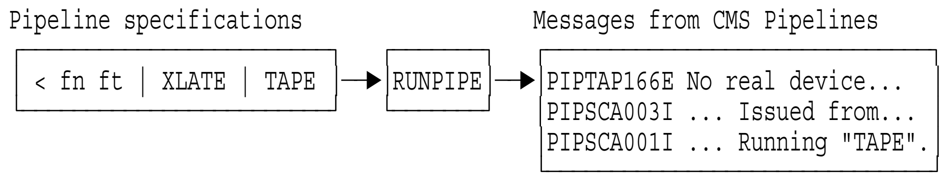
PIPCMD issues input records as pipeline commands. Typically the input consists of CALLPIPE pipeline commands that are built from data by a SPEC stage earlier in the pipeline.



```
CALLPIPE < fn ft | XLATE | *:
CALLPIPE LITERAL abc| TAPE
OUTPUT Hello!
```

```
Contents of "fn ft"
Hello!
```

RUNPIPE issues pipeline specifications in the same way that the PIPE command does. *CMS Pipelines* messages issued while the pipeline is running are written to the output stream rather than to the terminal.



```
< fn ft | XLATE | TAPE
```

```
PIPTAP166E No real device...
PIPSCA003I ... Issued from...
PIPSCA001I ... Running "TAPE".
```

Index