Advanced Pipelines
VM 3D3

Will J. Roden, Jr.

IBM
Endicott, NY

05/05/00

Disclaimer

Pipelines

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environment may vary significantly. Users of this document should verify the applicable data for their specific environment.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

Permission is hereby granted to publish an exact copy of this paper. IBM retains the title to the copyright in this paper as well as title to the copyright in all underlying works. IBM retinas the right to make derivative works ands to republish and distribute this paper to whomever it chooses in any way it chooses.

Agenda
Pipelines

- Juxtapose and Getfiles
- Predselect
- Append / Preface
- Callpipe
  - Handling Multiple Files
  - Strip Blanks
  - Reconnection
  - Substitute Many Lines for One
  - Addpipe
- Commit levels

JUXTAPOSE
Pipelines

- JUXTAPOSE
  - Prefixes records in the secondary stream with the most recently read record from the primary stream
  - Example:

Input: `JUX FILE`

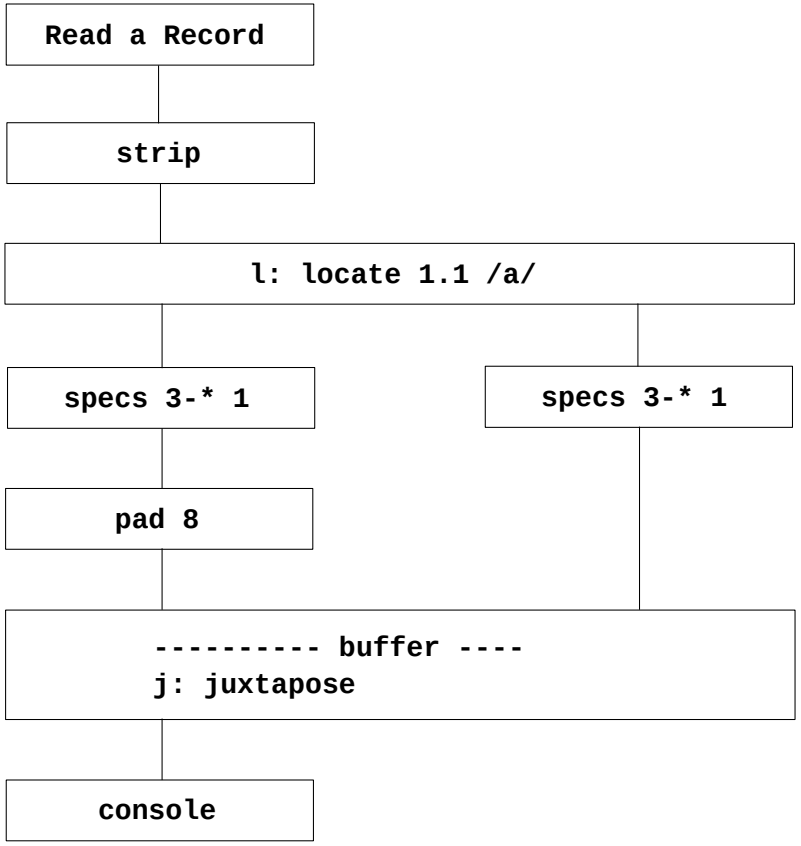
```
a John
j Master Plumber
a Will
h Fireman
h EMT
```

Output:

```
John   Master Plumber
Will   Fireman
Will   EMT
```

JUXTAPOSE ...
Pipelines

- JUXTAPOSE



JUXTAPOSE
Pipelines

- Code:

```
pipe (end ?)
     < jux file
  |  strip
  |l:locate 1.1 /a/
  |  specs 3-* 1
  |  pad 8
  |j:juxtapose
  |  console
  ?l:
  |  specs 3-* 1
  |j:
```

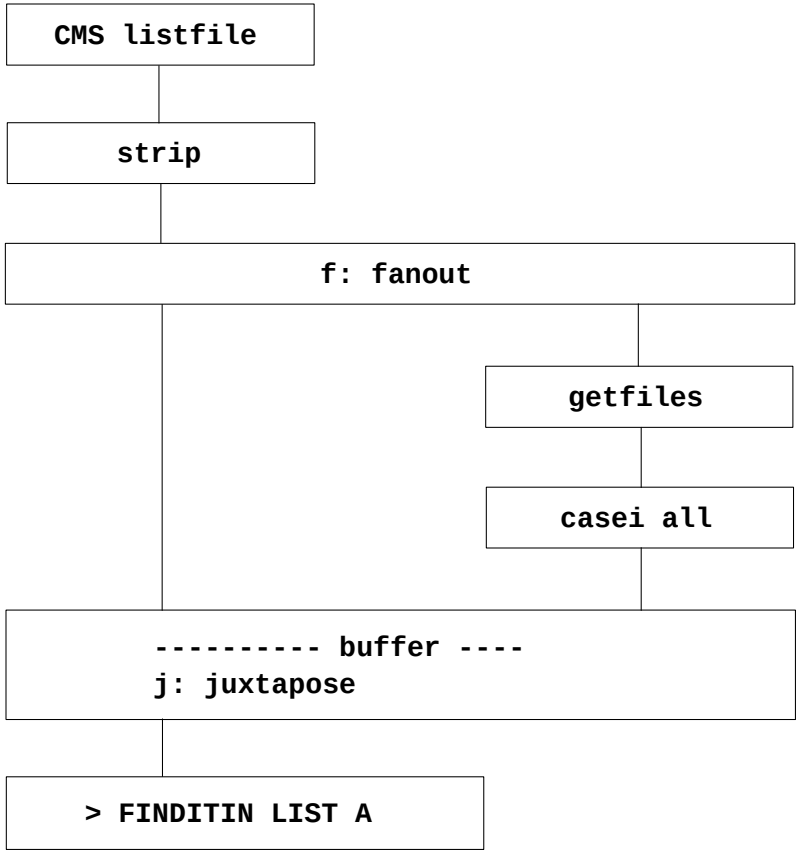
JUXTAPOSE and GETFILE Pipelines

- **Find It In**
  - **To find a string(s) in specified files**
  - **Example:**

```
finditin * exec a /will/!/john/
```

JUXTAPOSE and GETFILE ... Pipelines

- **JUXTAPOSE**



JUXTAPOSE and GETFILE Pipelines

- Find It In
  - To find a string(s) in specified files
  - Example:

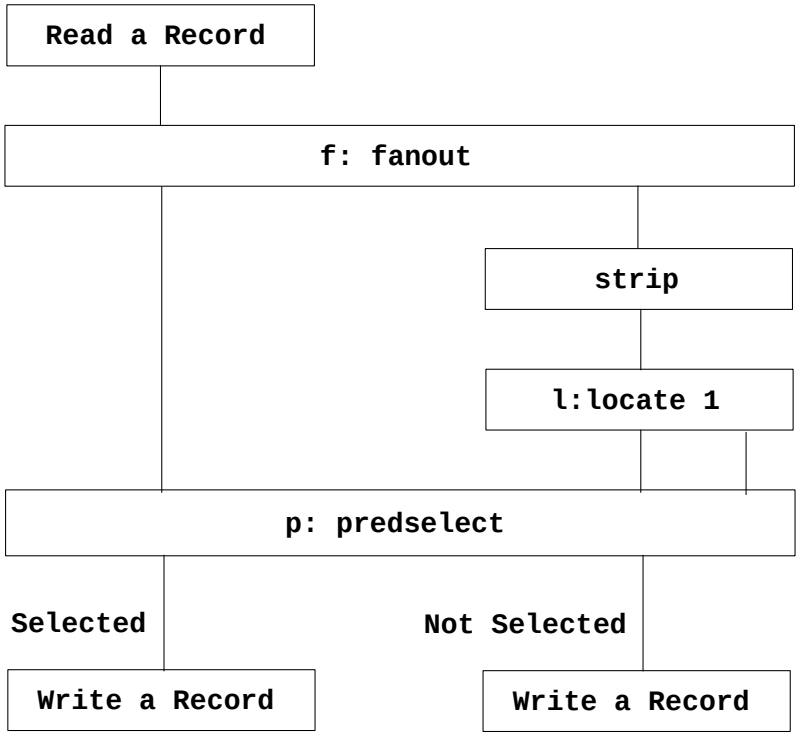
```
finditin * exec a /will/!/john/

arg fn ft fm lookfor
'Pipe (endchar ? name FINDITIN)',
 '  cms listfile' fn ft fm,
 '|f:fanout',
 '|j:juxtapose',
 '| > finditin list a',
 '?f:',
 '| getfiles',
 '| casei all' lookfor,
 '|j:',
```

PREDSELECT Pipelines

- PREDSELECT
  - Copies a record from its primary input stream to either its primary or secondary output stream depending on the order of arrival of input records on its other streams.
  - Example:
    - Discard all blank records without changing any other records. (leave their trailing blanks)

PREDSELECT ... Pipelines

- PREDSELECT



PREDSELECT Pipelines

- Example:

```
/* Discard blank lines.  */

PIPE (endchar ?)
  ’   < my input’,           /* Read input      */
  ’|f:fanout’,               /* Copy record     */
  ’|p:predselect’,              /* Keep original   */
  ’| > selected output a’,   /* Write a file    */
  ’?’,                       /* Second Pipeline */
  ’ f:’,                        /* Copy of record  */
  ’| strip’,                 /* Strip blanks    */
  ’|l:locate 1’,             /* Is length >= 1? */
  ’|p:’,                        /* Second input    */
  ’| > notsel output a’,     /* Write a file    */
  ’?’,                       /* Third Pipeline  */
  ’ l:’,                        /* Rcds length < 1 */
  ’|p:’                      /* Third input     */
```

Append / Preface Pipelines

- Preface runs its Pipeline argument first
- Append runs its Pipeline argument after EOF is received
- Watch for stage separator conflicts
  - Scanned twice ...
  - | is stage separator
  - \ is pipeline end character
  - Example: append literal one two three||split
- Uses
  - Keeps LITERALs in order
  - Allows reading multiple files
  - Allows running of entire pipelines
  - Allows one pipeline to be run after another has finished
    - ...| hole | append ...
  - Allows a pipeline to be run after several others have finished
    - ...|h:hole | append ... ?...|h: ?...|h:

Callpipe: Handling Multiple Files Pipelines

- Count the number of .foil lines in any list of files
- One way this can be run

```
'Pipe (endchar ! name FOILSCNT )',
'  cms listfile * foils a',
'|f:fanout',
'|j:juxtapose',
'|console',
'?f:',
'|foilscnt',
'|j:'
```

- A possible output

```
ADVPIPE FOILS   A1 22
B       FOILS   A1 41
EX      FOILS   A1 7
LISTOF  FOILS   A1 0
MICRO   FOILS   A1 16
```

Callpipe: Handling Multiple Files ... Pipelines

- REXX Solution

```
/* FOILSCNT REXX */
'PEEKTO' fileID          /* Get fileID  */
  do while rc = 0        /* Data avail? */
    'callpipe (endchar ?)',
     ' <' fileId,
     '| casei strfind /.foil/',
     '| count lines',
     '| *:'
    'READTO'             /* Consume     */
    'PEEKTO' fileID      /* Get fileID */
  end /* do while rc = 0 */
```

Callpipe: Handling Multiple Files ... Pipelines

- SPECS and PIPCMD Solution

```
'Pipe (endchar ? name FOILSCNT )',
  '  cms listfile * foils a',
  '|f:fanout',
  '| pad 21',
  '|j:juxtapose',
  '| console',
  '?f:',
  '| specs @callpipe < @ 1 1-* n',
     '@|| casei strfind /.foil/@ nw',
     '@|| count lines@ n',
     '@||*:@n',
  '| pipcmd',
  '|j:'
```

Callpipe: Strip Blanks Pipelines

- Eliminate Duplicate Blanks from each line

- One Way... But has a maximum and leaves a leading blank

```
|  change /  / /
|  change /  / /
|  change /  / /
```

- Another Way ... But assumes no x25 in text

```
|  spec 1-* 1 x25 n
|  split
|  joincont not trailing x25 / /
```

Callpipe: Strip Blanks ... Pipelines

- A Callpipe that works for a line - assumes no | or ? in text

```
callpipe
   literal .....
  |split
  |join * / /
  |*:
```

- The final code

```
'Pipe (endchar ?)',
  '|  append literal   Line    one  is here.',
  '|  append literal This is the     second.',
  '|  append literal          The third.    ',
  '|  specs @callpipe@ 1',
    '@   literal@ n 1-* nw',
    '@|| split@ n',
    '@|| join * / /@ n',
    '@||*:@ n',
  '|  pipcmd',
  '|  console,
```

- The Output

```
Line one is here.
This is the second.
The third.
```

Callpipe: Reconnection Pipelines

- Reconnect Do heading then body ...
  - ► The EXEC

```
/* RECONN EXEC */
'Pipe (endchar ?)',
  '   < input data',
  '|  reconn',
  '|  console'
```

  - ► The REXX

```
/* RECONN REXX */
'callpipe',
  ' *:',
  '|  take 4',
  '|*:'
'callpipe',
  ' *:',
  '|  locate 5.1 /4/',
  '|*:'
```

Callpipe: Reconnection ... Pipelines

- INPUT DATA

```
Nbr S Last     Ans   Next     Nx Dpt
    E Actopm   Code  Action   Ac Nxt
    V Done           To Do    ID Act
----+----1----+----2----+----3----+-
001 3 RECEIVED       ANSWER   KL A32
002 3 RECEIVED       ANSWER   BR A32
003 3 RECEIVED       ANSWER   BI A00
004 5 ANSWERED COLD  INT1     VM B29
005 3 ANSWERED DUP   CLOSE    CO B26
006 3 OPENED         RECEIVE  BI A00
007 4 REROUTED       RECEIVE  PE B18
008 3 OPENED         RECEIVE  BI A00
009 4 RECEIVED       ANSWER   BE B18
010 3 ANSWERED COLD  SSV      DE A31
011 4 CLOSED   CNEW  NONE
012 4 CLOSED   CNEW  NONE
013 3 OPENED         RECEIVE  CP A64
014 4 RECEIVED       ANSWER   NE B18
015 1 OPENED         RECEIVE  CP A64
016 3 RECEIVED       ANSWER   XX A31
```

Callpipe: Reconnection ... Pipelines

- The Output

```
Nbr S Last     Ans   Next     Nx Dpt
    E Actoon   Code  Action   Ac Nxt
    V Done           To Do    ID Act
----+----1----+----2----+----3----+-
007 4 REROUTED       RECEIVE  PE B18
009 4 RECEIVED       ANSWER   BE B18
011 4 CLOSED   CNEW  NONE
012 4 CLOSED   CNEW  NONE
014 4 RECEIVED       ANSWER   NE B18
```

Callpipe: One line into Many Pipelines

- LANGUAGE HTML

```
<html>
<head>
<title>Language Selection</title>
</head>
<body>
<FORM METHOD=POST ACTION="sendfile.cgi?roden">
<INPUT TYPE="submit" VALUE="Submit">
<h5>Your Favorite Programming Language:</h5>
<select name="language">
  <option>
</select>
</FORM>
</body>
</html>
```

- OPTION LIST

```
APL
Assembler
C
C++
JAVA
Pipelines
REXX
```

Callpipe: One line into Many ... Pipelines

- The Output

```
<html>
<head>
<title>Language Selection</title>
</head>
<body>
<FORM METHOD=POST ACTION="sendfile.cgi?roden">
<INPUT TYPE="submit" VALUE="Submit">
<h5>Your Favorite Programming Language:</h5>
<select name="language">
  <option>APL
  <option>Assembler
  <option>C
  <option>C++
  <option>JAVA
  <option selected>Pipelines
  <option>REXX
</select>
</FORM>
</body>
</html>
```

Callpipe: One line into Many ... Pipelines

- The Code

```
'Pipe (end ?)',
  '   < language html',
  '|l:locate @<option>@',
  '|  spec @callpipe < option list||*:@1',
  '|  pipcmd',
  '|  spec /    <option>/ 1 1-* n',
  '|f:faninany',
  '|  > language htm a',
  '?l:',
  '|f:'
```

Addpipe Pipelines

- Differences between callpipe and addpipe
  - ► Callpipe runs as a "pipeline subroutine"
    - – When callpipe terminates, connections are put back
  - ► Addpipe runs as a "separate task"
    - – When addpipe terminates, connections are terminated
- Nine ways addpipe can be used:

  addpipe B | C
  addpipe B | C |*.input:
  addpipe *.output: | B | C
  addpipe *.input: | B | C
  addpipe B | C |*.output:
  addpipe *.input: | B | C |*.input:
  addpipe *.output: | B | C |*.output:
  addpipe *.input: | B | C |*.output:
  addpipe *.output: | B | C |*.input:

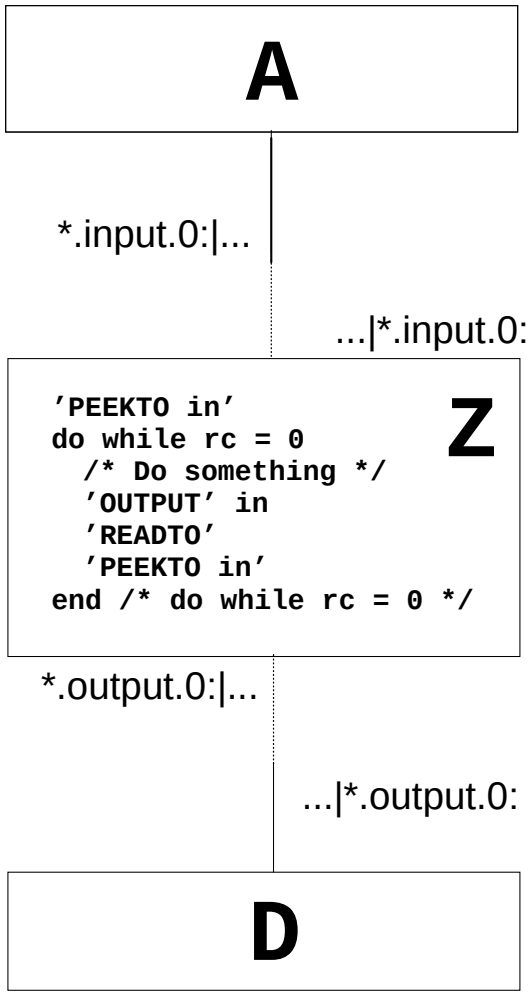
Addpipe's Nine Cases Pipelines

- Pipeline A | Z | D

- Z -- a typical REXX stage

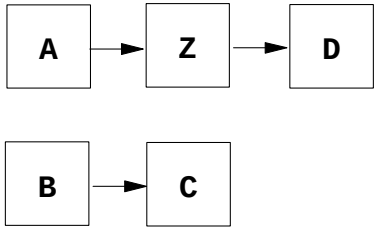
```
'PEEKTO in'
do while rc = 0
  /* Do something */
  'OUTPUT' in
  'READTO'
  'PEEKTO in'
end /* do while rc = 0 */
```

Addpipe's Nine Cases — Pipelines

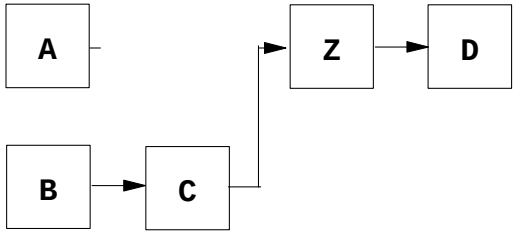


Addpipe's Nine Cases ... — Pipelines

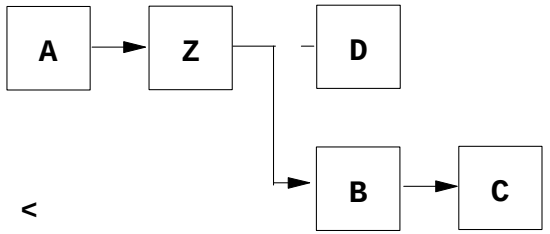
- **addpipe B | C**



- **addpipe B | C |*.input:**

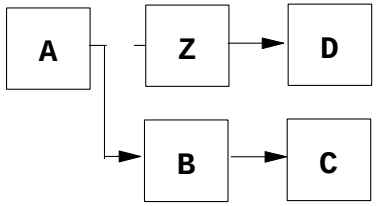


- **addpipe *.output: | B | C**

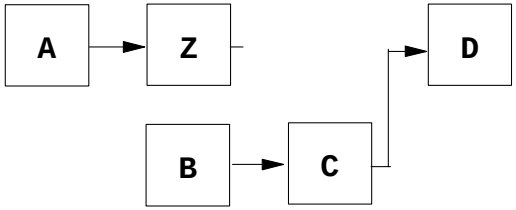


Addpipe's Nine Cases ... Pipelines

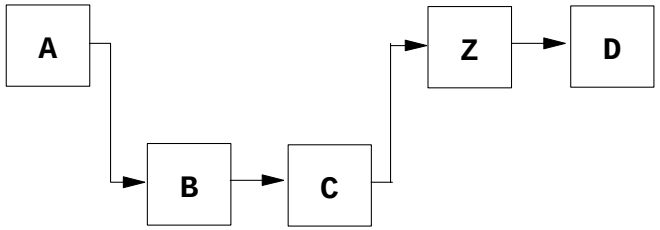
- addpipe *.input: | B | C



- addpipe B | C |*.output:

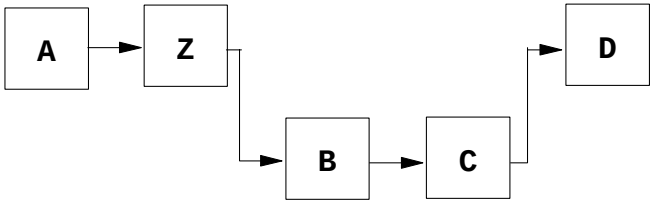


- addpipe *.input: | B | C |*.input:

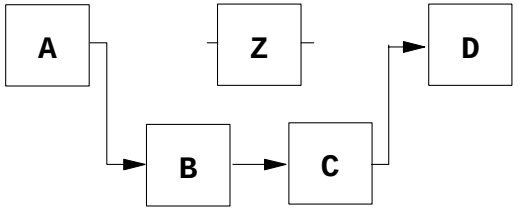


Addpipe's Nine Cases ... Pipelines

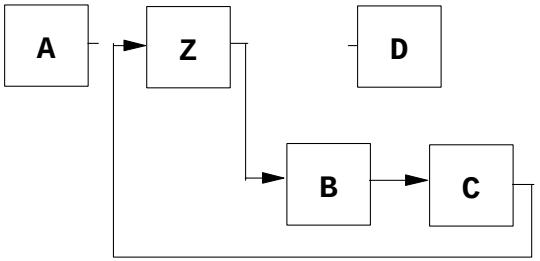
- addpipe *.output: | B | C |*.output:



- addpipe *.input: | B | C |*.output:

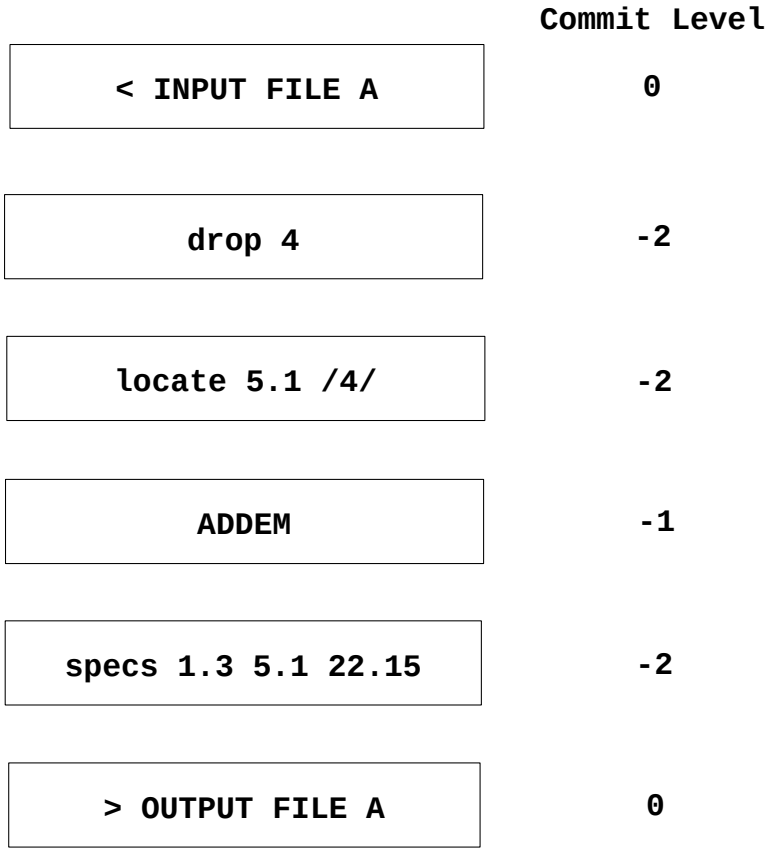


- addpipe *.output: | B | C |*.input:



Commit levels Pipelines

The commit level allows stages to coordinate their initial execution. This assists in preventing the loss of data from a device drives such as reading an A3 file.



For More Information: Pipelines

- CMS Pipelines User's Guide - SC24-5777
- CMS Pipelines Reference - SC24-5778
- CMS Pipelines Author's Edition - SL26-0018
- Quick Reference - SX24-5290
- Development Contact
  - Will J. Roden, Jr.
    - Internet:: roden@us.ibm.com
    - Phone: (607) 752-5065
    - Web: http://www.vm.ibm.com/devpages/roden
    - Mail: IBM Department G37G
    - Mail: 1701 North Street
    - Mail: Endicott,NY 13760