How Pipelines Changed My Life – Or At Least My Tooling in z/VM

David Kreuter

dkreuter@vm-resources.com

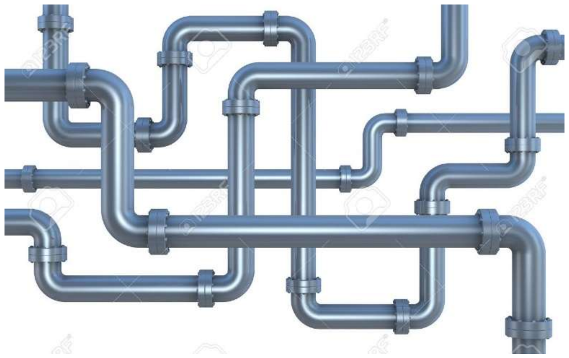
VM RESOURCES LTD
Software * Consulting * Training

How Pipelines Changed My Life – Or At Least My Tooling in z/VM

Abstract: This presentation will show the value of using PIPELINES in z/VM as a productivity tool and for efficient use of sysprog time in z/VM. PIPELINEs is a must in building, managing, and administering z/VM systems. Adding PIPEs knowledge to your skill set is well worth learning and a must for tooling on z/VM.

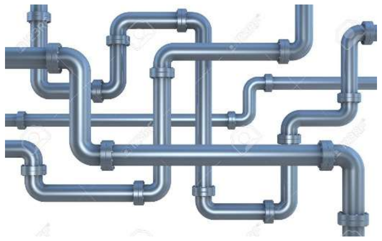
PIPELINES as a Productivity Tool

- Learning PIPE basics great productivity aid.
- Capturing CP and CMS command good place to start.
- As valuable as REXX.
- PIPE and REXX together an unbeatable pair.



PIPELINES – what you will learn today

- Terminology
- PIPE Flow
- Constructing productivity aids with PIPELINEs
- Stages, filters, device drivers.
- Issuing CP command and filtering in PIPELINEs
- Issuing CMS commands and filtering in PIPELINEs
- Reading and Writing CMS files



PIPELINES

- PIPELINES is the PIPE program with a set of specifications. The specifications are one or more programs known as stages.
- Stages are separated by the vertical bar. There is a connection from the output stream from the stage to the left of the vertical bar to the input stream of the stage to the right of the vertical bar.
- Data is passed left to right through a pipeline.
- Stages run independently co-ordinated by the pipeline dispatcher.
- Stages obtain data from the dispatcher or host interface and produce to data to a host interface or to the dispatcher.
- Categories of stages.

Useful Terms and Definitions

- A stage is a program point in a pipeline.
- An identifier for a stage that defines multiple data streams, one for each occurrence of the label in a pipeline specification.
- Stream is data passing through a pipe (input or output).
- A record is a unit being passed on a stream
- A label on a stage identifier that defines multiple data streams.
- Labeled stage denotes multistream
- Standalone label stage connects to a stream
- A connector is at the beginning or ending of a pipelines indicating how the pipeline is to be connected to the to streams in the stage defining the pipeline.

Useful Terms and Definitions

- Label
- Character string at the beginning of a stage.
- Mixed case.
- End with a colon.
- **Declared** when used for the first time in a pipeline specification. Precedes a stage. Defines the primary streams.
- **Referenced** in a label stage. No further arguments allowed. Defines the secondary and subsequent streams.
- *Examples:*
- CHRC: CP
- CHRC: W.IN1:

- Connector:
- A full connector consists of an asterisk, a period, a keyword indicating a direction, a period, a stream identifier, and a colon
- *Examples:*
- *:
- *.output:
- *.input:

Host Interface and Device Driver Stages

- **Issue CP commands and capture results in the pipelines**
- **Issue CMS commands and capture results in the pipeline**
- **Read and write CMS files.**
- Place and retrieve data into/from the CMS stack
- Read and write to the CONSOLE
- Manage the 3270
- Interface to spool devices
- IUCV services
- TCPIP sockets
- Many more host interfaces
- Fetch and write to REXX variables including stem variables.
- Attractive replacement to EXECIO! Live the EXECIO free way!

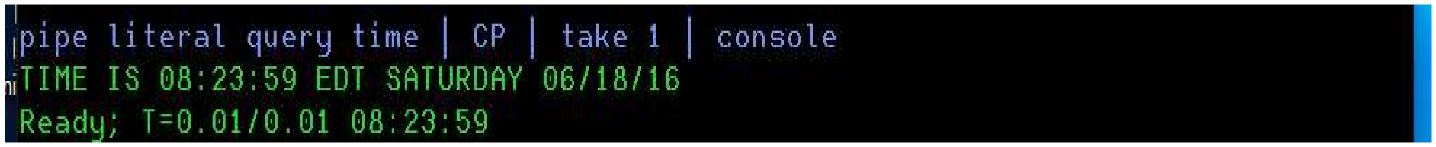
Filter and Selection Stages

Perform a function on the data within the PIPELINE.

- Chop, strip, count, snake, sort, locate, nlocate, find, specs, pad, split, reverse, change, between, inside, frlabel, tolabel, not, merge, take, drop, duplicate, deal, tokenize, unique, pack, unpack,
- …. and much more.

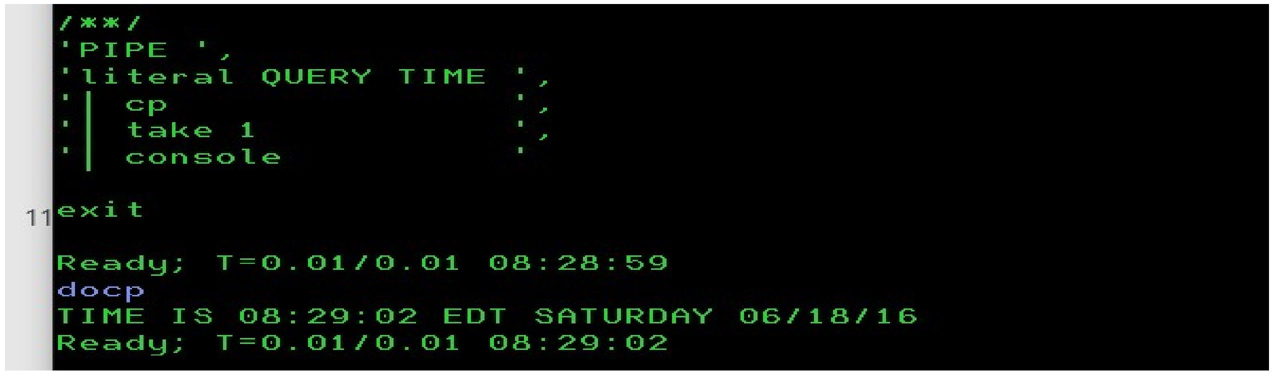
Places to Execute Your PIPE

- CONSOLE



```
pipe literal query time | CP | take 1 | console
TIME IS 08:23:59 EDT SATURDAY 06/18/16
Ready; T=0.01/0.01 08:23:59
```

- REXX Container:



```
/**/
'PIPE ',
'literal QUERY TIME ',
'|  cp               ',
'|  take 1           ',
'|  console          '

exit

Ready; T=0.01/0.01 08:28:59
docp
TIME IS 08:29:02 EDT SATURDAY 06/18/16
Ready; T=0.01/0.01 08:29:02
```

CONSOLE Stage

INPUT or OUTPUT DRIVER STAGE

- Read or write to the CONSOLE
- CONSOLE read when first stage
- “ First Stage” = read from console (VM READ state
- Writes to primary output stream if connected

```
''PIPE CONSOLE',
'| REVERSE ',
'| > DEMO2 SAMPLE A'

demo2
forward
Ready; T=0.01/0.01
16:16:15
```

'First stage' Console ended when null string entered or the EOF String is found

```
type demo2
sample


drawrof
```

CONSOLE Stage

INPUT or OUTPUT DRIVER STAGE

- Read or write to the CONSOLE
- CONSOLE read when first stage
- Both! … positional ...

```
demo3:

'PIPE ',
'CONSOLE EOF /END!/',
'| CONSOLE'
```

Read from the CONSOLE then echo to console, end when END! entered

```
demo3
help
help
aide
aide
END!
```

CONSOLE Stage

INPUT or OUTPUT DRIVER STAGE

- Read or write to the CONSOLE
- CONSOLE read when first stage other times CONSOLE write
- “Not a First Stage” = reads from primary input stream and writes to the console. Writes to primary output stream if connected.

```
'PIPE',
'LITERAL Hello world',
'| CONSOLE '

Hello world
```

Gotta start somewhere!

```
/**/
Say 'Hello
world'

Hello world
```

LITERAL stage

Producer

- LITERAL <string>
- Writes a character string as a record to the primary output stream if connected.
- If records in primary input stream copied to primary output stream *after* character string

```
pipe literal QUERY SET|CONSOLE
QUERY SET
Ready; T=0.01/0.01 14:08:34
```

```
pipe literal hello world | literal query set | console
query set
hello world
Ready; T=0.01/0.01 14:10:26
```

CP *<string>*

Device driver

- Issue CP command
- String if provided is a CP command.
- Streams:
- Primary input stream contains CP commands to be issued.
- Primary output stream contains **result** of commands.
- Secondary output stream contain the return code. – later!

```
pipe cp query set| console
MSG ON   , WNG ON   , EMSG ON   , ACNT OFF, RUN OFF
LINEDIT ON , TIMER OFF , ISAM OFF, ECMODE ON
ASSIST OFF            , PAGEX OFF, AUTOPOLL OFF
IMSG ON   , SMSG OFF , AFFINITY NONE    , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF    , STMULTI OFF   00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XA , SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON, 370ACCOM OFF, TIMEBOMB IDLE
```

CP

Device Driver

- Issue CP command
- Streams:
- Primary input stream contains CP commands to be issued.
- Primary output stream contains result of commands.
- Secondary output stream contain the return code. – later!

```
pipe literal query set | cp |console
MSG ON   , WNG ON   , EMSG ON   , ACNT OFF, RUN OFF
LINEDIT ON , TIMER OFF , ISAM OFF, ECMODE ON
ASSIST OFF           , PAGEX OFF, AUTOPOLL OFF
IMSG ON   , SMSG OFF , AFFINITY NONE   , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF    , STMULTI OFF   00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XA , SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON, 370ACCOM OFF, TIMEBOMB IDLE
```

CP and elementary PIPE think

Device driver and filter

```
pipe cp query set | console
MSG ON   , WNG ON   , EMSG ON   , ACNT OFF, RUN OFF
LINEDIT ON , TIMER OFF , ISAM OFF, ECMODE ON
ASSIST OFF           , PAGEX OFF, AUTOPOLL OFF
IMSG ON   , SMSG OFF , AFFINITY NONE   , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF    , STMULTI OFF   00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XA , SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON, 370ACCOM OFF, TIMEBOMB IDLE
```

```
pipe cp query set | locate /MACHINE/ | Console
MACHINE XA , SVC76 CP, NOPDATA OFF, IOASSIST OFF
```

```
pipe cp query set | split , | locate /MACHINE/ | console
MACHINE XA
```

CP with stage string and input stream

Device driver and filter

```
pipe literal query time | literal Query userid exp|cp indicate|cons
AVGPROC-000% 01
MDC READS-000001/SEC WRITES-000001/SEC HIT RATIO-100%
PAGING-1/SEC STEAL-000%
Q0-00000(00000)                           DORMANT-00004
Q1-00000(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00000(00000) EXPAN-001 E3-00000(00000)
PROC 0000-000% IFL
LIMITED-00000
MAINT    AT VMR1
TIME IS 13:01:24 PST FRIDAY 04/29/2011
CONNECT= 00:13:03 VIRTCPU= 000:00.02 TOTCPU= 000:00.04
```

Flow: When string present on CP stage issue first then process the input stream.

Recently needed to do this…

Take output of CP QUERY DASD and display RDEV and VOLSER

```
query dasd
DASD 0300 CP SYSTEM LI0300   0
DASD 0302 CP SYSTEM DS0302   0
:
DASD 231B CP SYSTEM LI231B   0
DASD 231C CP SYSTEM LI231C   0
Ready; T=0.01/0.01 09:48:54
```

```
0300 LI0300
0302 DS0302
:
231B LI231B
231C LI231C
```

Recently needed to do this…

Take output of CP QUERY DASD and display RDEV and VOLSER

```
pipe literal query dasd | cp |specs w2 1 w5 nw | console
0300 LI0300
0302 DS0302
:
231B LI231B
231C LI231C
Ready; T=0.01/0.01 09:55:15
```

SPECS stage

SPECS: A Most Powerful Stage!

Powerful Stage

- Powerful stage.
- Practically a programming language!
- Output records created from:
  - Input records contents
  - Constants
  - Internal data

SPECS stage

- Multiple input and output streams
- Can compute numeric expressions, compare data, maintains counters, data conversions and can do conditionals

SPECS: elements

Powerful Stage

- The basics:
  - ***Eye ko ooh ah:***
- Input
- Conversion
- Output
- Alignment

SPECS stage

SPECS: eye ko ooh ah (ICOA)

Powerful Stage

Basic specs: Input Conversion Output Alignment

```
type cities list a
Austin
Seattle
Boston
Kansas City
Toronto
pipe < cities list a
| specs /Cities:/ 1 1-* strip nw /with SHARE conferences?/ nw
|console
Cities: Austin with SHARE conferences?
Cities: Seattle with SHARE conferences?
Cities: Boston with SHARE conferences?
Cities: Kansas City with SHARE conferences?
Cities: Toronto with SHARE conferences?
```

***Eye ko ooh ah*?**

Input
Conversion
Output
Alignment

```
specs  1-*  nw.15 center 1-* c2x nw.26
```

Powerful Stage

```
pipe < cities list a|specs  1-*  nw.15 center 1-* c2x nw.26|console
  Austin      C1A4A2A3899540404040404040
  Seattle     E28581A3A39385404040404040
  Boston      C296A2A3969540404040404040
Kansas City   D28195A281A240C389A3A84040
  Toronto     E396999695A396404040404040
```

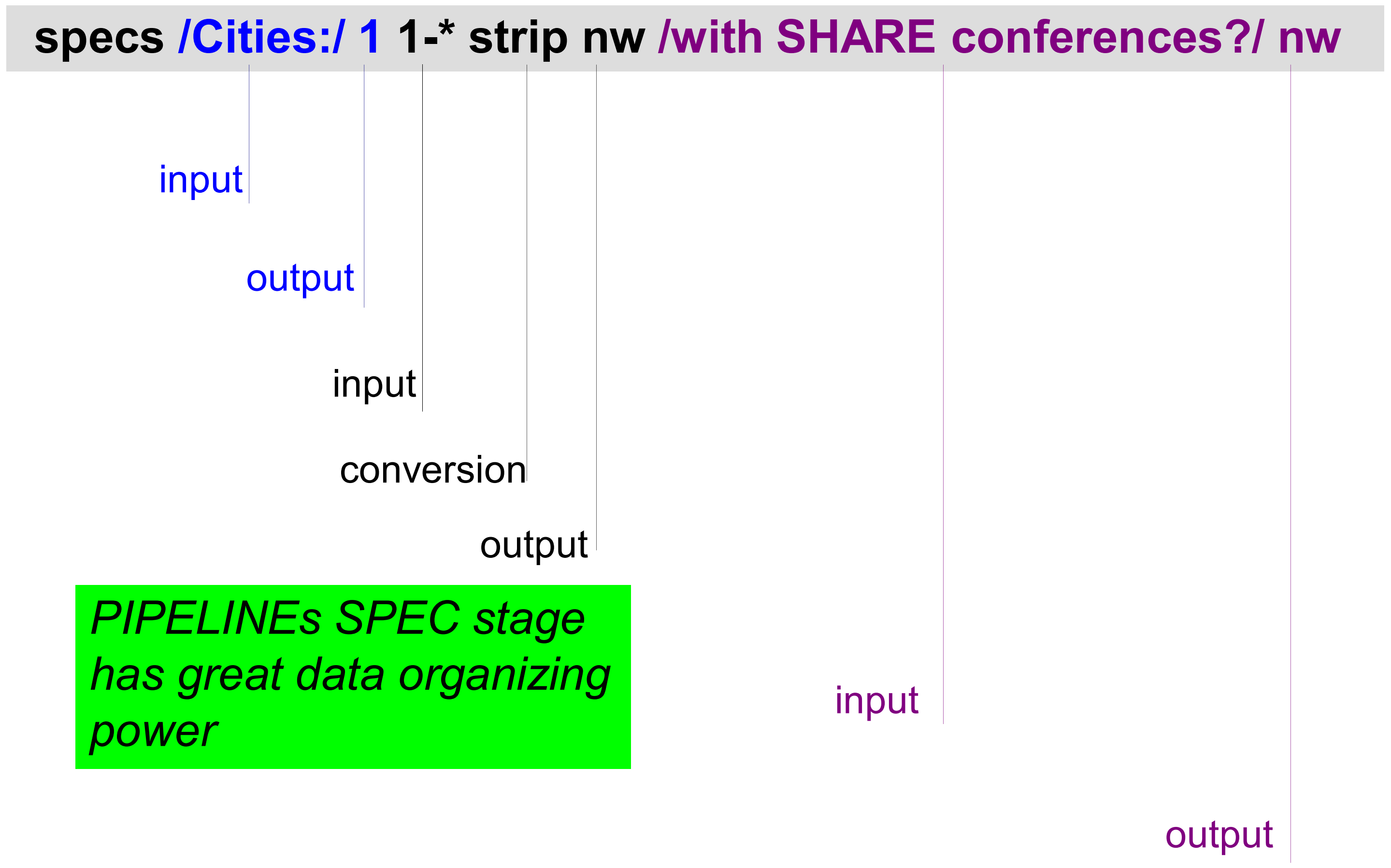
input

conversion

output

Eye ko ooh ah?

Input
Conversion
Output
Alignment



```
specs /Cities:/ 1 1-* strip nw /with SHARE conferences?/ nw
```

PIPELINEs SPEC stage has great data organizing power

SPECS: words

Powerful Stage

```
specs w2 1 w5 nw
```

Take output of CP QUERY DASD and display RDEV and VOLSER. "take the 2nd word of input record place in column 1 of the output and take word 5 of the input records and place in the next blank delimited position in the output record"

```
pipe literal query dasd | cp |specs w2 1 w5 nw | console
0300 LI0300
0302 DS0302
:
231B LI231B
231C LI231C
Ready; T=0.01/0.01 09:55:15
```

Specification list using words from the input. Column number used for output positions.

Add a title with the PREFACE stage

```
pipe literal query dasd | cp|specs w2 1 w5 nw
| preface literal RDEV VOLSER
|cons
RDEV VOLSER
0300 LI0300
0302 DS0302
:
231B LI231B
231C LI231C
Ready; T=0.01/0.01 10:11:20
```

“Use the PREFACE stage to invoke another stage or subroutine pipeline, write the records produced by that stage or subroutine pipeline to its primary output stream, and then copy all records from its primary input stream to its primary output stream.”

"Use the PREFACE stage to invoke another stage or subroutine pipeline, write the records produced by that stage or subroutine pipeline to its primary output stream, and then copy all records from its primary input stream to its primary output stream."

```
pipe literal query dasd | cp|specs w2 1 w5 nw
| preface literal RDEV VOLSER|cons
RDEV VOLSER
0300 LI0300
0302 DS0302
:
231B LI231B
231C LI231C
Ready; T=0.01/0.01 10:11:20
```

The "PREFACE LITERAL RDEV VOLSER" stage puts the string RDEV VOLSER in the primary output stream. Then all the remaining records are copied unchanged to the primary output stream. So it is great for creating a title!

CMS <*string*>

Device driver

- Issue CMS command with full command resolution (start at implied EXEC search).
- String if provided is a CMS command.
- Streams:
  - Primary input stream contains CMS commands to be issued.
  - Primary output stream contains result of commands.
  - Secondary output stream contain the return code. – later!

```
pipe literal listfile D* EXEC A |CMS QUERY IMPEX |console
IMPEX    = ON
DEMO1    EXEC     A1
DEMO2    EXEC     A1
DEMO3    EXEC     A1
DOATT    EXEC     A1
DOC      EXEC     A1
Ready; T=0.01/0.01 13:57:20
```

CMS *with a wee bit of PIPE think*

Device driver and filter

- LOCATE filter with the “?” as the string delimiter.
- Quickly, routinely, and easily extend CMS command function!

```
pipe cms query search|locate ?R/O?|console
MNT190  190   S     R/O
MNT19E  19E   Y/S   R/O
MNT19D  19D   Z     R/O
```

COMMAND *<string>*

Device driver

- Issue CMS command with resolution similar to REXX's ADDRESS COMMAND
- String if provided is a CMS command.
- Streams:
  - Primary input stream contains CMS commands to be issued.
  - Primary output stream contains result of commands.
  - Secondary output stream contain the return code. – later!

```
pipe literal SENDFILE PROFILE EXEC TO *|COMMAND
Ready(-0003); T=0.01/0.01 13:53:35

pipe literal EXEC SENDFILE PROFILE EXEC TO *|COMMAND
Ready; T=0.01/0.01 13:53:58
```

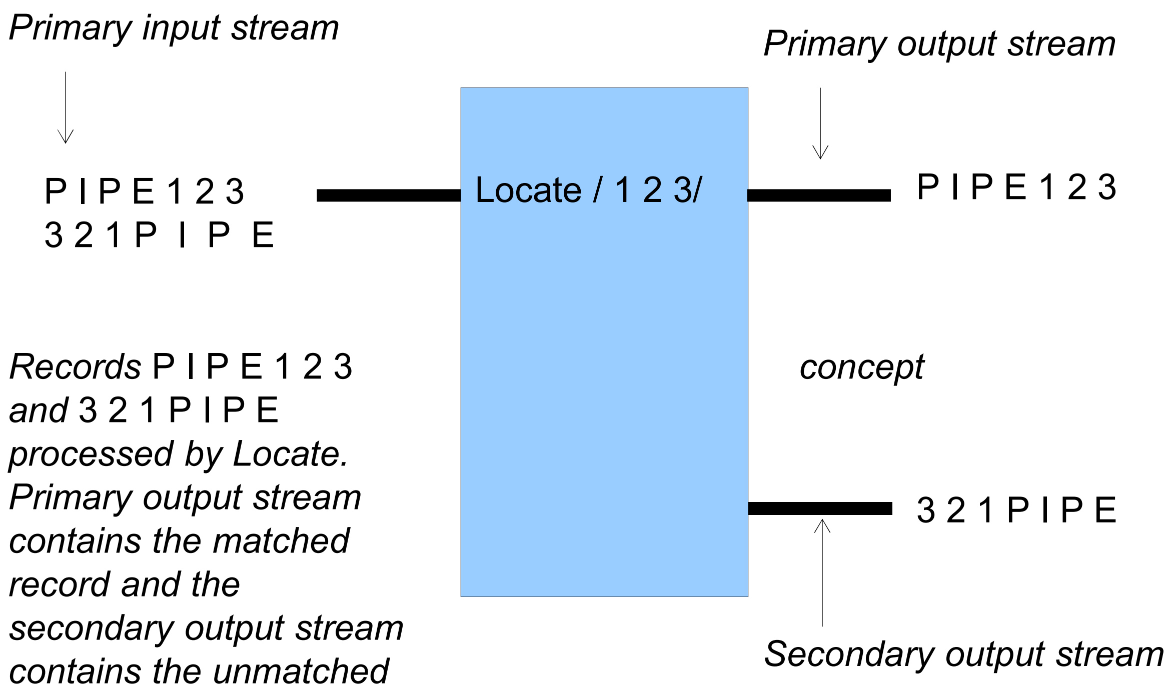
COMMAND *<string>*

Device driver

- Issue CMS command with resolution similar to REXX's ADDRESS COMMAND
- String if provided is a CMS command.
- Streams:
  - Primary input stream contains CMS commands to be issued.
  - Primary output stream contains result of commands.
  - Secondary output stream contain the return code. – later!

```
pipe literal QUERY IMPEX|COMMAND EXEC SENDFILE PROFILE EXEC TO * | console
File PROFILE EXEC A1 sent to * at VMR1 on 04/29/2011 14:03:31
IMPEX     = ON
Ready; T=0.01/0.01 14:03:31
```

PIPELINE CONCEPTUAL TOPOLOGY CASE STUDY #2: Multistream output

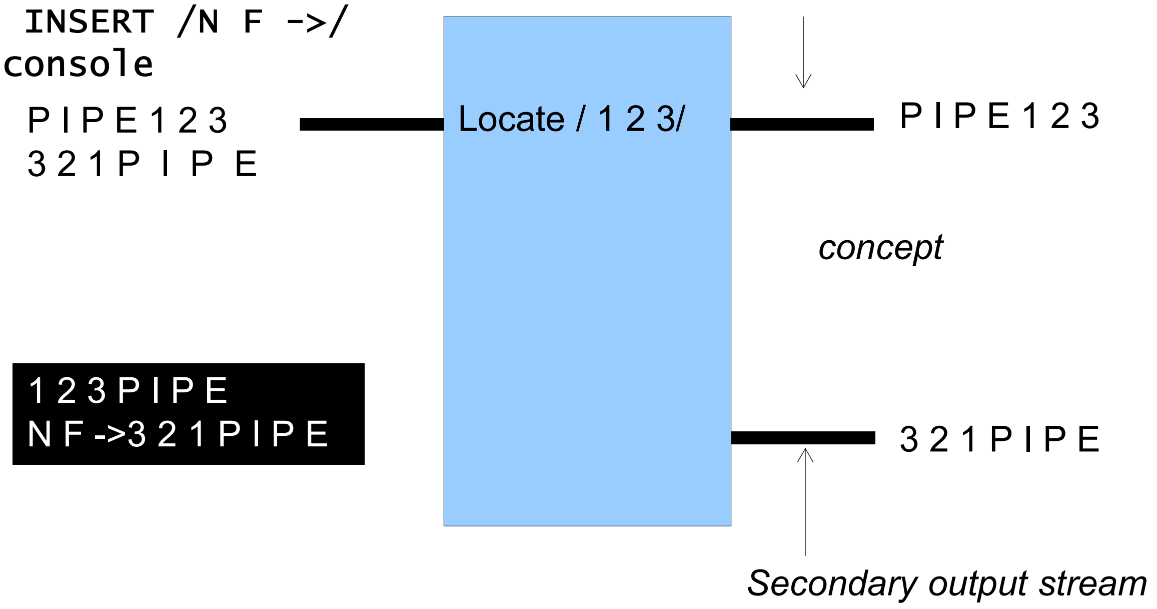


Records P I P E 1 2 3 *and* 3 2 1 P I P E *processed by Locate. Primary output stream contains the matched record and the secondary output stream contains the unmatched record.*

PIPELINE TOPOLOGY CASE STUDY #2: Multistream output: The PIPELINE

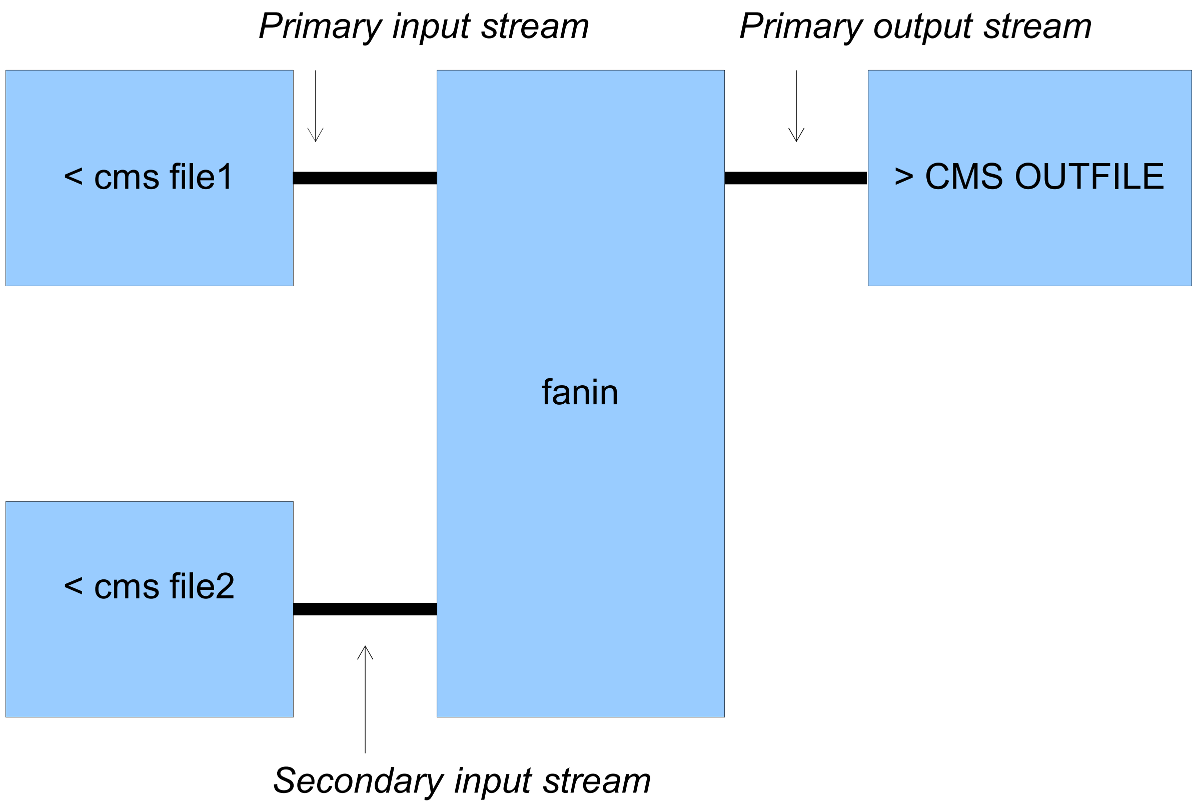
Primary input stream

```
pipe (end ?) ? literal 3 2 1 P I P E
|Literal 1 2 3 P I P E
| L: Locate /1 2 3/
|CONSOLE
? L:
| INSERT /N F ->/
|console
```

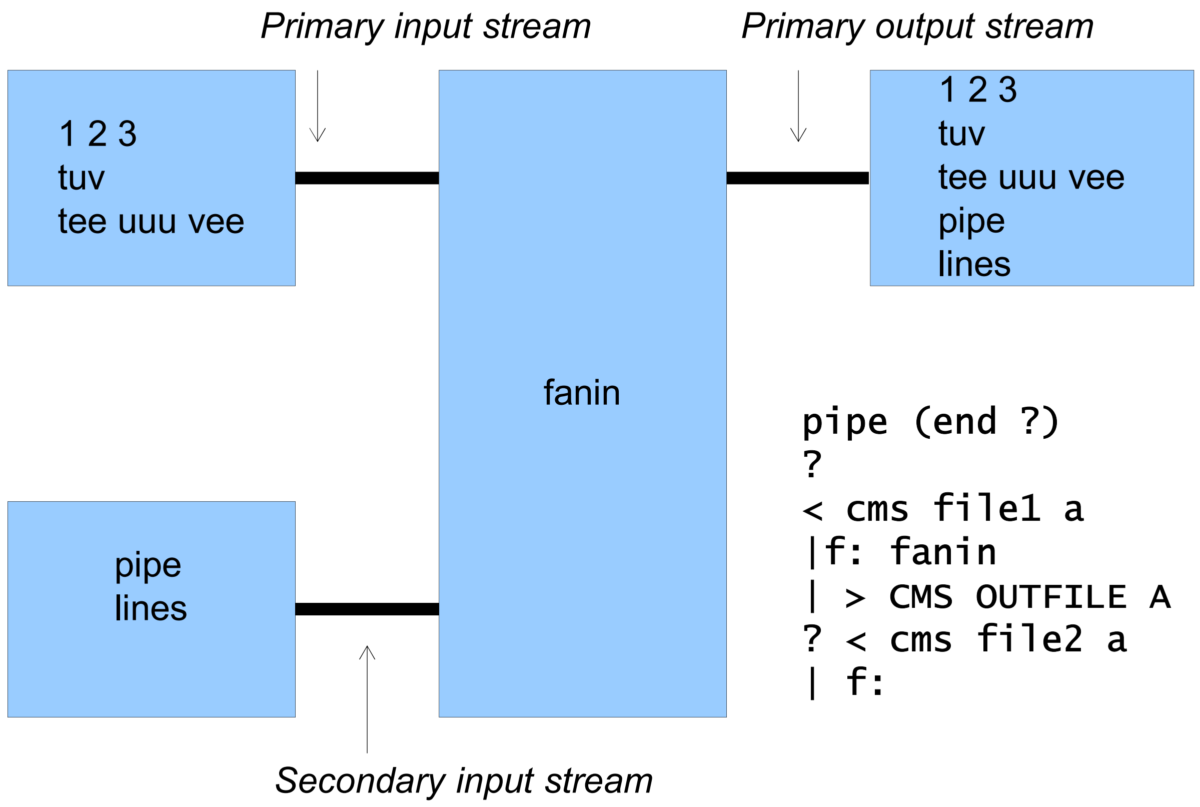


```
1 2 3 P I P E
N F ->3 2 1 P I P E
```

PIPELINE CONCEPTUAL TOPOLOGY
CASE STUDY #3: Multistream input



PIPELINE TOPOLOGY CASE STUDY #3: Multistream input: The PIPELINE



```
pipe (end ?)
?
< cms file1 a
|f: fanin
| > CMS OUTFILE A
? < cms file2 a
| f:
```

PIPELINE CONCEPTUAL TOPOLOGY CASE STUDY #4: Multistream input and output

```
USER LGLOPR NOLOG 32M 32M ABCDEG
 ACCOUNT ACT1 PROP
 MACH XA
 IPL 190
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
 LINK OPERATOR 191 291 RR
 MDISK 191 3390 1827 001 VMRRES  MR READ WRITE
USER TOOLS NOLOG
*MDISK 0191 3390 0000 End  VMR191 RR
 MDISK 0192 3390 0000 End  TOOLS  RR ALL SOME FEW
```

```
/* remembrance of things past */

PIPE (end ?),
?,
 < U D A,
| strip,
| L: LOCATE W1 /USER/,
| specs w2 1.10 ,
| J: JUXTAPOSE ,
| CONSOLE,
?,
L:,
| abbrev MDISK 2 ,
| J:
```

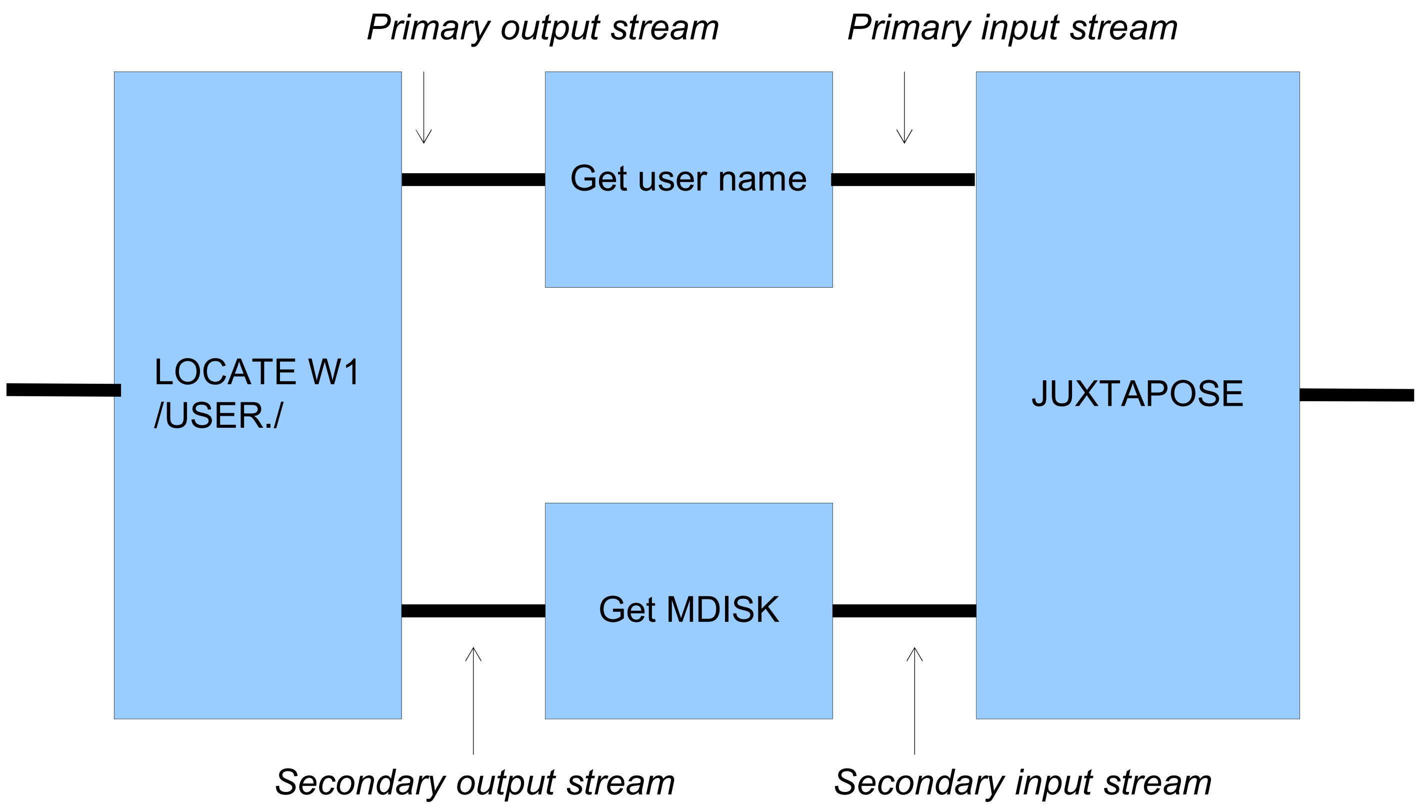
```
LGLOPR     MDISK 191 3390 1827 001 VMRRES  MR READ    WRITE
TOOLS      MDISK 0192 3390 0000 End  TOOLS  RR ALL SOME FEW
```

PIPELINE CONCEPTUAL TOPOLOGY CASE STUDY #4: Multistream input and output

```
/* remembrance of things past */

PIPE (end ?),
?,
 < U D A,
| strip,
| L: LOCATE W1 /USER/,
| specs w2 1.10 ,
| J: JUXTAPOSE ,
| CONSOLE,
?,
L:,
| abbrev MDISK 2 ,
| J:
```

PIPELINE CONCEPTUAL TOPOLOGY CASE STUDY #4: Multistream input and output



<

Device driver

- Read CMS files into the pipeline
- Uses the <mdsk driver for minidisk files
- Uses the <sfs driver for shared file system objects

```
pipe < my favs a | console
Charlie Parker
John Coltrane
Miles Davis
Charles Gayle
Sirone
Artie Shaw
```

GETFILES

Device driver

- Read a list of CMS files into the pipeline

```
pipe command LISTFILE * TEAMS
|getfiles
|console
Buffalo
New York
Miami
Jacksonville
Toronto
New York
Boston
Los Angeles
Miami
Cleveland
Toronto
New York
```

GETFILES

Device driver

- Read a list of CMS files into the pipeline

```
pipe command LISTFILE * TEAMS
|console
|getfiles
|console
FOOTBALL TEAMS    A1
Buffalo
New York
Miami
Jacksonville
BASEBALL TEAMS    A1
Toronto
New York
Boston
Los Angeles
BASKET   TEAMS    A1
Miami
Cleveland
Toronto
New York
```

COUNT:

Filter

- Filter that counts items in records from the input stream: bytes, words, lines, minline and maxline. Choose one, more, or all options.
- With one output stream records on input stream are discarded.
- When secondary output stream connected count is written to secondary output stream and the primary output stream contains the records.
- Record containing counts written to primary output stream
- Count record is delayed until eof.

```
pipe < my favs a
| count bytes words lines
| console
480 11 6
```

COUNT: multistream

Filter

- Filter that counts items in records from the input stream: bytes, words, lines, minline and maxline. Choose one, more, or all options.
- With one output stream records on input stream are discarded.
- When secondary output stream connected count is written to secondary output stream and the primary output stream contains the records.
- Record containing counts written to primary output stream

```
pipe (endchar ?) ? < MY FAVS A
| C: COUNT bytes words lines
| console
?
C:
| INSERT /==>/
|CONSOLE
```

```
Charlie Parker
John Coltrane
Miles Davis
Charles Gayle
Sirone
Artie Shaw
==>480 11 6
```

APPEND stage: introduce records from a second file!

Control

- Outputs data from the device driver specified as operand on output stream **after** Data is shorted. Device driver can certainly be a “first stage” command.
- May be used to insert data from multiple files in the same specification.

```
pipe < my favs a
| append < some composer a
| cons
```

```
Charlie Parker
John Coltrane
Miles Davis
Charles Gayle
Sirone
Artie Shaw
Beethoven
Stravinsky
Elgar
Mozart
```

PREFACE stage: introduce records from a second file!

Control

- Outputs data from the device driver specified as operand on output stream **before** data is shorted. Device driver can certainly be a “first stage” command.
- May be used to insert data from multiple files in the same specification.

```
pipe < my favs a
| preface < some composer a
| cons
```

```
Beethoven
Stravinsky
Elgar
Mozart
Charlie Parker
John Coltrane
Miles Davis
Charles Gayle
Sirone
Artie Shaw
```

APPEND stage

Control

- Outputs data from the device driver specified as operand on output stream **after** Data is shorted. Device driver can certainly be a “first stage” command.
- May be used to insert data from multiple files files in the same specification.

```
pipe < my favs a
| append literal Jazz rules!
|console
```

```
Charlie Parker
John Coltrane
Miles Davis
Charles Gayle
Sirone
Artie Shaw
Jazz rules!
```

>

Device driver

- Create or replace a CMS file
- Uses the <mdsk driver for minidisk files
- Uses the <sfs driver for shared file system objects

```
pipe cp query time |> CPQT LISTING A
```

```
TIME IS 17:38:46 EDT TUESDAY 05/10/11
CONNECT= 30:18:23 VIRTCPU= 000:02.09 TOTCPU= 000:02.34
```

```
pipe cp query time |> CPQT LISTING A
```

replaces-->

```
TIME IS 17:41:45 EDT TUESDAY 05/10/11
CONNECT= 30:21:23 VIRTCPU= 000:02.09 TOTCPU= 000:02.35
```

>>

Device driver

- Create or Append to a CMS file
- Uses the <mdsk driver for minidisk files
- Uses the <sfs driver for shared file system objects

```
pipe cp query time | >> CPQTV1 LISTING A
```

```
TIME IS 17:46:36 EDT TUESDAY 05/10/11
CONNECT= 30:26:14 VIRTCPU= 000:02.09 TOTCPU= 000:02.35
```

```
pipe cp query time | >> CPQTV1 LISTING A
```

Appends -->

```
TIME IS 17:46:36 EDT TUESDAY 05/10/11
CONNECT= 30:26:14 VIRTCPU= 000:02.09 TOTCPU= 000:02.35
TIME IS 17:49:14 EDT TUESDAY 05/10/11
CONNECT= 30:28:52 VIRTCPU= 000:02.09 TOTCPU= 000:02.36
```

LOCATE

Filter

- Selects records with a matching string
- All matching records sent to output stream
- Revisit multistream

```
query dasd
DASD 0123 CP OWNED   VMRRES    50
DASD 0124 CP SYSTEM VMRW01    47
DASD 0125 CP SYSTEM VMRW02    1
DASD 0126 CP OWNED   VMRSPL    1
DASD 0127 CP OWNED   VMRPAG    1
DASD 0128 CP SYSTEM VMRWRK    1
DASD 0192 CP SYSTEM TOOLS     1
DASD 0500 CP SYSTEM MAI500    1
DASD 1127 CP SYSTEM VMRNEW    0
DASD F123 CP SYSTEM VMRVMR    0
```

```
pipe CP QUERY DASD
| LOCATE /OWNED/
| CONSOLE
DASD 0123 CP OWNED   VMRRES    50
DASD 0126 CP OWNED   VMRSPL    1
DASD 0127 CP OWNED   VMRPAG    1
```

SPLIT

Filter

- Splits record at specified data or location
- May increase records to output

```
pipe literal a b c d e f g|console|split|console
a b c d e f g
a
b
c
d
e
f
g
```

Default is AT BLANK

CHANGE

Filter

- Change contents of input records based on matching string within the specified column range. Unchanged records sent to secondary output stream if connected otherwise to primary output stream.
- Revisit multistreams

```
pipe < listof ips a
|cons
192.168.0.xxx
10.1.1.xxx
1.2.xxx.3
192.168.150.xx
xxx.yyy.xxx/24
```

```
pipe < listof ips a
|change /xxx/100/
|cons
192.168.0.100
10.1.1.100
1.2.100.3
192.168.150.xx
100.yyy.100/24
```

Change 'xxx' to '100' anywhere in the input record.

SORT

Filter

- Not a First stage – sort input records

```
Abercrombie, Neil, Hawaii, 1st
Acevedo-Vila, Anibal, Puerto Rico, At Large
Ackerman, Gary, New York, 5th
Aderholt, Robert, Alabama, 4th
Alexander, Rodney, Louisiana, 5th
Allen, Tom, Maine, 1st
Akin, Todd, Missouri, 2nd
Andrews,Robert E., New Jersey, 1st
Baca, Joe, California, 43rd
Bachus, Spencer, Alabama, 6th
Baird, Brian, Washington, 3rd
Baker, Richard, Louisiana, 6th
Baldwin,Tammy, Wisconsin, 2nd
Ballance, Frank, North Carolina, 1st
Ballenger, Cass, North Carolina, 10th
Barrett,J.Gresham, South Carolina, 3rd
Bartlett, Roscoe, Maryland, 6th
Barton, Joe, Texas, 6th
Bass, Charles, New Hampshire, 2nd
Beauprez, Bob, Colorado, 7th
```

Sort by state or territory (field 3)

```
pipe < members listall *
| sort fs , f3
| TAKE 20',
| CONSOLE
```

SORT

Filter

- Not a First stage – sort input records

```
pipe < members listall *
| sort fs , f3
| TAKE 20',
| CONSOLE
Aderholt, Robert, Alabama, 4th
Bachus, Spencer, Alabama, 6th
Bonner, Jo, Alabama, 1st
Cramer, Robert E. "Bud", Alabama, 5th
Davis,Artur, Alabama, 7th
Everett,Terry, Alabama, 2nd
Rogers, Mike, Alabama, 3rd
Young, Don, Alaska, At Large
Faleomavaega, Eni F. H., American Samoa
Flake, Jeff, Arizona, 6th
Franks, Trent, Arizona, 2nd
Grijalva, Raul, Arizona, 7th
Hayworth, J.D., Arizona, 5th
Kolbe, Jim, Arizona, 8th
Pastor, Ed, Arizona, 4th
```

Sorted by field 3 (field separator is the comma)

TAKE

Filter

- TAKEs “n” records from beginning or end of input stream.
- Revisit multistreams
- May output less records

```
type jazz favs

Miles Davis
John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
Sirone
Artie Shaw
```

```
pipe < jazz favs a
|take 2
|cons
Miles Davis
John Coltrane

pipe < jazz favs a
|take last 3
|cons
Thelonius Monk
Sirone
Artie Shaw
```

XLATE

Filter

- XLATE replaces characters according to a translate table.
- Useful for proper collating of hexadecimal devices addresses

```
pipe literal 1 A 2 B
| split
|sort
|cons
A
B
1
2
```

WRONG

```
pipe literal 1 A 2 B
|split
|xlate A-F FA-FF
|sort|xlate FA-FF A-F
|console
1
2
A
B
```

CORRECT

XLATE

Filter

- XLATE replaces characters according to a translate table.
- Useful for proper collating of hexadecimal devices addresses

XLATE A-F FA-FF changes hex character to be larger then 0-9

XLATE FA-FF A-F changes it back!

```
pipe literal 1 A 2 B
|split
|xlate A-F FA-FF
|sort|xlate FA-FF A-F
|console
1
2
A
B
```

CORRECT

STRIP

Filter

- Removes leading and/or trailing characters or strings.
- Default is to remove leading and trailing blanks.

```
pipe
 literal one
|literal   two
|literal         David
|console
      David
  two
one
```

```
pipe
 literal one
|literal   two
|literal         David
|strip
|console
David
two
one
```

DROP

Filter

- DROPs “n” records from beginning or end of input stream.
- Revisit multistreams
- May output less records

```
pipe < jazz favs
|console
Miles Davis
John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
Sirone
Artie Shaw
```

```
pipe < jazz favs
|drop 2
|console
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
Sirone
Artie Shaw
```

DROP

Filter

- DROPs “n” records from beginning or end of input stream.
- Revisit multistreams
- May output less records

```
pipe < jazz favs
|console
Miles Davis
John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
Sirone
Artie Shaw
```

```
pipe < jazz favs
|drop last 2
|console
Miles Davis
John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
```

DROP multistream

Filter

- DROPs "n" records from beginning or end of input stream.
- Secondary output stream contains dropped records

```
pipe (endchar %)
< jazz favs
|V: drop 2
|console
%
V:
|insert /dropped: /
|cons
dropped: Miles Davis
dropped: John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
Sirone
Artie Shaw
```

```
pipe (endchar %)
< jazz favs
|V: drop last 2
|console
%
V:
|insert /dropped: /
|cons
Miles Davis
John Coltrane
Charles Gayle
Charlie Parker
Bill Evans
Thelonius Monk
dropped: Sirone
dropped: Artie Shaw
```

SORT

Filter

- Not a First stage – sort input records

```
Abercrombie, Neil, Hawaii, 1st
Acevedo-Vila, Anibal, Puerto Rico, At Large
Ackerman, Gary, New York, 5th
Aderholt, Robert, Alabama, 4th
Alexander, Rodney, Louisiana, 5th
Allen, Tom, Maine, 1st
Akin, Todd, Missouri, 2nd
Andrews,Robert E., New Jersey, 1st
Baca, Joe, California, 43rd
Bachus, Spencer, Alabama, 6th
Baird, Brian, Washington, 3rd
Baker, Richard, Louisiana, 6th
Baldwin,Tammy, Wisconsin, 2nd
Ballance, Frank, North Carolina, 1st
Ballenger, Cass, North Carolina, 10th
Barrett,J.Gresham, South Carolina, 3rd
Bartlett, Roscoe, Maryland, 6th
Barton, Joe, Texas, 6th
Bass, Charles, New Hampshire, 2nd
Beauprez, Bob, Colorado, 7th
```

Sort by state or territory (field 3)

```
pipe < members listall *
| sort fs , f3
| TAKE 20',
| CONSOLE
```

CP and multistream

Driver and multistream

- Primary output stream contains command results
- Secondary output stream contains return code

```
pipe literal SET LMSG ON
| CP
Ready(00003); T=0.01/0.01 05:20:54

pipe (endchar %) literal SET LMSG ON
|RC: CP
%
RC:
| Specs /Return code:/ 1 1-* nw
| CONS
Return code: 3
Ready; T=0.01/0.01 05:21:33

pipe (endchar %) literal SET  MSG ON
|RC: CP
%
RC:
| Specs /Return code:/ 1 1-* nw
| CONS
Return code: 0
Ready; T=0.01/0.01 05:21:49
```

The CP driver supports multistreams. The return code is passed to the secondary output stream. What stage(s) would be useful to capture the return code if you were in a REXX environment?

CMS and multistream

Driver and multistream

- Primary output stream contains command results
- Secondary output stream contains return code

```
pipe (endchar %) literal L WHAT EVER A
|RC: CMS
%
RC:
| Specs /Return code:/ 1 1-* nw
| CONS
Return code: 28
Ready; T=0.01/0.01 05:29:23

pipe (endchar %) literal L WHAT EVER A
|RC: CMS
|CONS
%
RC:
| Specs /Return code:/ 1 1-* nw
| CONS
DMSLST002E File not found
Return code: 28
Ready; T=0.01/0.01 05:29:40
```

The CMS (and COMMAND) driver supports multistreams. The return code is passed to the secondary output stream.

CMS and multistream

Driver and multistream

- Primary output stream contains command results
- Secondary output stream contains return code

```
pipe (endchar %)
literal L WHAT EVER A
|CMS
|CONS
DMSLST002E File not found
Ready(00028); T=0.01/0.01 05:31:21

pipe (endchar %)
literal L WHAT EVER A
|CMS
Ready(00028); T=0.01/0.01 05:35:53
```

The CMS driver with a single output stream. Final return code is reflected on PIPE exit.

LOCATE, FANIN, and ELASTIC

Filter

- Organize the output nicer! Use the FANIN and ELASTIC stages.

```
pipe (endchar ?) ? cp query dasd |L: locate /OWNED/
|F: FANIN |console
?
L: |insert /--->/ |ELASTIC |F:
```

```
DASD 0123 CP OWNED  VMRRES   50
DASD 0126 CP OWNED  VMRSPL   1
DASD 0127 CP OWNED  VMRPAG   1
--->DASD 0124 CP SYSTEM VMRW01   47
--->DASD 0125 CP SYSTEM VMRW02   1
--->DASD 0128 CP SYSTEM VMRWRK   1
--->DASD 0192 CP SYSTEM TOOLS    1
--->DASD 0500 CP SYSTEM MAI500   1
--->DASD 1127 CP SYSTEM VMRNEW   0
--->DASD F123 CP SYSTEM VMRVMR   0
```

FANIN combine multiple input streams into a single output stream. Can use more than two input streams.
ELASTIC buffers enough records to avoid a stall. Can use single or multiple input streams.

LOCATE and ELASTIC

Filter

```
pipe (endchar ?) ?
cp query dasd|L: locate /OWNED/|E: ELASTIC|console
? L:
|insert /--->/ |ELASTIC |E:
```

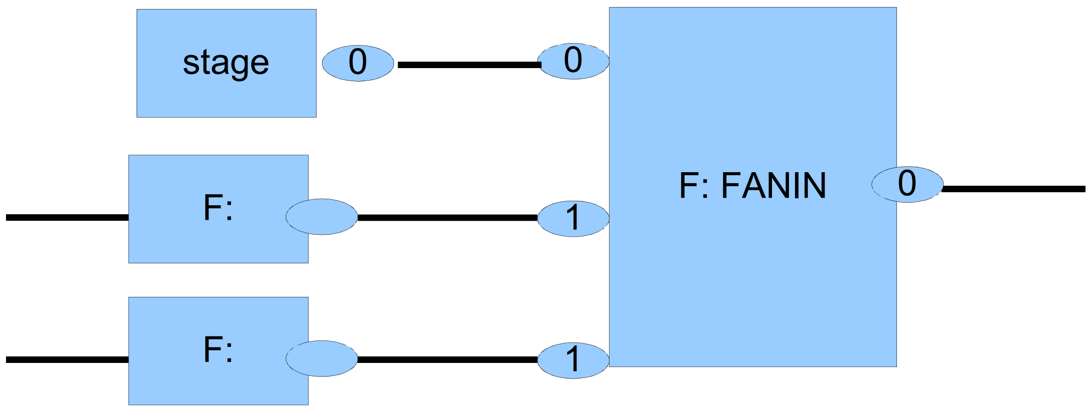
```
DASD 0123 CP OWNED  VMRRES   50
DASD 0126 CP OWNED  VMRSPL   1
DASD 0127 CP OWNED  VMRPAG   1
--->DASD 0124 CP SYSTEM VMRW01   47
--->DASD 0125 CP SYSTEM VMRW02   1
--->DASD 0128 CP SYSTEM VMRWRK   1
--->DASD 0192 CP SYSTEM TOOLS    1
--->DASD 0500 CP SYSTEM MAI500   1
--->DASD 1127 CP SYSTEM VMRNEW   0
--->DASD F123 CP SYSTEM VMRVMR   0
```

ELASTIC with two input streams copies primary input stream records to its output stream and buffers secondary input stream records. When EOF reached on primary output stream it starts writing from the buffer to the primary output stream, continuing to buffer secondary input stream records.

multistream

FANIN topolgy

- Consider the topology.
- FANIN supports multiple input streams.
- Has one output stream
- “drains” input stream in consecutive order unless streams specified as parameters



multistream

FANIN: read multiple files

- Consider the topology.
- Drains input stream in order.

```
ONE  POTATO
one
   potato
```

```
TWO  POTATOES
TWO
   SPUDS
```

```
COOK  METHODS
boil
fry
mash
bake
```

```
pipe (Endc ?)
? < one potato|SPUDS: FANIN|CONS
? < two potatoes a|SPUDS:
? < cook methods a|SPUDS:
one
   potato
TWO
   SPUDS
boil
fry
mash
bake
```

multistream

FANIN: read multiple files

- Consider the topology.
- Drains input stream in order unless instructed otherwise

```
ONE  POTATO
one
   potato
```

```
TWO  POTATOES
TWO
   SPUDS
```

```
COOK  METHODS
boil
fry
mash
bake
```

```
pipe (Endc ?)
? < one potato|SPUDS: FANIN 2 0 1
|CONS
? < two potatoes a|SPUDS:
? < cook methods a|SPUDS:
boil
fry
mash
bake
one
   potato
TWO
   SPUDS
```

Multistream

JUXTAPOSE

- Preface part of a record on other records
- Record Breaks
- Input stream:

•Primary: as it becomes available and is a one record buffer.

•Secondary: records read as available and are prefixed with the buffered primary input stream record.

- Output stream:

•Primary output stream contains combined record

JUXTAPOSE

Multistream

```
type instru ments

Instrument Piano
  Bud Powell
  Thelonius Monk
  Cecil Taylor
  Bill Evans
Instrument Saxophone
  Sonny Rollins
  John Coltrane
  Charels Gayle
  Charlie Parker
  Evan Parker
Instrument Bass
  Sirone
  William Parker
  Charles Mingus
  Paul Chambers
  Reggie Workman
Instrument Drums
  Art Blakey
  Max Roach
  Dannie Richmond
  Michael Wimberly
  Hamid Drake
```

```
type juxta1 exec

/**/

'PIPE (endchar \)',
'< INSTRU MENTS A',
'|L: LOCATE /Instru/',
'|specs w2 1 ',
'| J: JUXTAPOSE ',
'| CONSOLE',
'\',
'L:',
'|J:'
```

Show on the same record the instrument each jazz great plays!

```
juxta1
Piano   Bud Powell
Piano   Thelonius Monk
Piano   Cecil Taylor
Piano   Bill Evans
Saxophone   Sonny Rollins
Saxophone   John Coltrane
Saxophone   Charels Gayle
Saxophone   Charlie Parker
Saxophone   Evan Parker
Bass   Sirone
Bass   William Parker
Bass   Charles Mingus
Bass   Paul Chambers
Bass   Reggie Workman
Drums   Art Blakey
Drums   Max Roach
Drums   Dannie Richmond
Drums   Michael Wimberly
Drums   Hamid Drake
```

JUXTAPOSE

Multistream

```
type juxta1 exec

/**/

'PIPE (endchar \)',
'< INSTRU MENTS A',

'|L: LOCATE /Instru/',

'|specs w2 1 ',



'| J: JUXTAPOSE ',


'| CONSOLE',

'\',
'L:',


'|J:'
```

Read INSTRU MENTS

Send matches to primary output others to secondary output

Place word 2 in column 1 of output record (1 record buffer)

Preface secondary input record with instrument name

2nd pipe: records that do not contain instru

Standalone label – output records on secondary input stream to JUXTAPOSE

JUXTAPOSE Case Study

Multistream

Case study:
Write a multistream pipeline the puts the user name in front of each record of a directory entry. Output should look like:

```
AVSVM    USER AVSVM NOLOG 32M 64M G 64
AVSVM INCLUDE IBMDFLT
AVSVM ACCOUNT  1   AVSVM
:
AVSVM MDISK 191 3390 2256 003 VMRW01  MR RAVS
AVSVM*
TSAFVM   USER TSAFVM NOLOG 16M 16M G
TSAFVM INCLUDE IBMDFLT
TSAFVM ACCOUNT 1 TSAFVM
TSAFVM OPTION MAXCONN 256 COMSRV DIAG98 ACCT
TSAFVM MACH XA
TSAFVM IUCV ANY
```

Use the JUXTAPOSE stage of course. Consider using FANOUT, FANIN, LOCATE ….

PIPELINES – next session

- LABs
- Fun with PIPELINEs
- CP and CMS command management.
- See you after a break!

How Pipelines Changed My Life – Or At Least My Tooling in z/VM

Abstract: This presentation will show the value of using PIPELINES in z/VM as a productivity tool and for efficient use of sysprog time in z/VM. PIPELINEs is a must in building, managing, and administering z/VM systems. Adding PIPEs knowledge to your skill set is well worth learning and a must for tooling on z/VM.