VM/ESA & VSE/ESA Technical Conference
May 31- June 3, 2000
Orlando, Florida

Intro to CMS Pipelines Lab

Sessions M50, M51

Chuck Morse
IBM Washington Systems Center
morsec@us.ibm.com

RETURN TO INDEX

Disclaimer

Trademarks

The following are trademarks of International Business Machines Corporation. Those identified with an (*) are registered trademarks of International Business Machines

- IBM*
- S/390
- VM/ESA*

The following are trademarks of the corporations identified

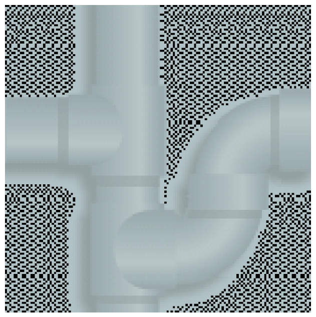
UNIX - is a registered trademark, licensed exclusively by X/Open Company, Ltd.

Agenda

- The basics
- Getting data into and out of a pipe
- Selecting specific records
- Altering the contents of records
- Putting pipeline stages in subroutines
- Simple multi-stream pipes

What is a Pipeline?

- Created by John Hartmann, IBM Denmark
- Modeled after, but far exceeds the capabilities of, UNIX pipes
- Consists of a series of programs, called stages, through which data flows
- Allows complex problems to be processed by a series of simple programs
- Requires a whole new way of looking at problems - "Pipethink"



CMS Pipelines

- Can be invoked
  - from the command line
  - from a REXX exec
- Stages are delimited by a stage separator
  - Default stage separator character is the vertical bar |
- Each stage processes the data and passes it on to the next stage

Our First Example

- On the command line

```
pipe literal When you wish upon a star | console
```

- In a REXX Exec

```
/* This is example 1 */
'pipe',
   'literal When you wish upon a star |',
   'console';
```

Two Very Useful Stages

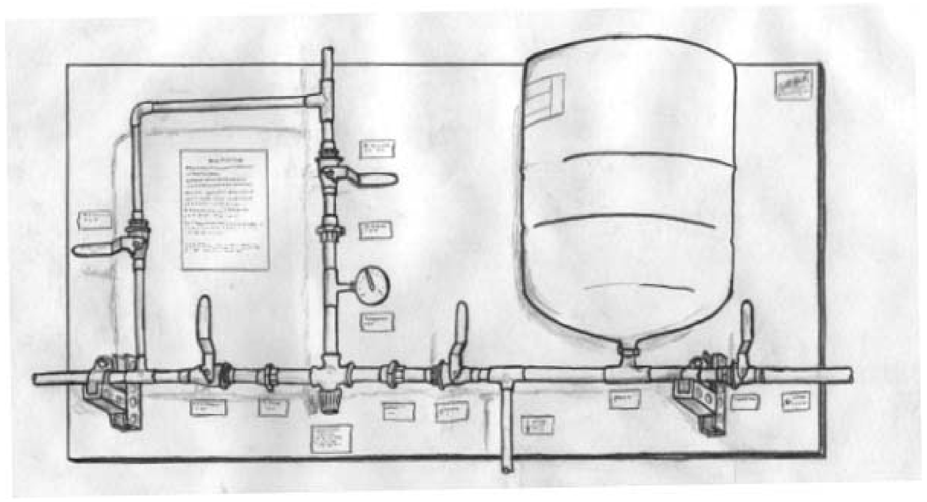
- Two stages for getting help
  - **`help`**
    - Provides standard help information for all stages
  - **`ahelp`** (Author’s Help)
    - In many cases provides more extensive help information and examples
- Syntax is basically the same for each stage
    - **`pipe help/ahelp`** - displays help for last error message
    - **`pipe help/ahelp menu`** - displays list of stages, etc.
    - **`Pipe help/ahelp console`** - displays help for console stage

Exercise #2

1. View the help or ahelp information for the following stages
   - help
   - ahelp
   - literal
   - console
2. View the help or ahelp menu of stages
3. Issue the help or ahelp stage with no options
4. Issue the help or ahelp stage with no options (again)

Types of Pipeline Stages

- Device Drivers
- Selection Stages
- Transformation Stages
- Others
  - Gateways
  - Control stages



Device Drivers

- Interface with the outside world
- Used to get data into and out of the pipeline
  - Input stages
    - Usually used at the beginning of a pipe to get data into the pipe
  - Output stages
    - Usually used later in the pipe to take data out of the pipe
    - When not at the end of a pipe, they also pass their data to the next stage
  - Some stages can be both input or output stages, depending on their placement in the pipe

Some Input Device Drivers

- `literal` - inserts a string into the pipe
- `< fn ft fm` - reads a disk file
- `reader` - reads from the virtual card reader
- `state/statew` - puts file information into the pipe

Must be first stage in the pipeline

Some Output Device Drivers

- **`> fn ft fm`** - writes a new disk file or rewrites an existing one
- **`>> fn ft fm`** - appends to an existing disk file or writes a new one
- **`punch`** - writes records to the virtual punch
- **`printmc`** - writes records to the virtual printer

Must not be first stage in the pipeline

Some Device Drivers Do Both

- Act as input device driver if first stage
- Act as output device driver if not first stage
- Examples
  - **`console`** - reads from or writes to the console
  - **`var`** - retrieves or sets REXX variables
  - **`stem`** - retrieves or sets REXX stem variables
  - **`stack`** - reads or writes the program stack
  - **`tape`** - reads or writes tape

Some Other Device Drivers

- Host command processors
  - cp - executes a CP command
  - cms - executes a CMS command
  - Response is passed through the pipe to the next stage
  - Command can be
    - passed as a parameter
      - `pipe cp q n | console`
    - passed through the pipe from the previous stage
      - `pipe literal q n | cp | console`

Input Stages Later in the Pipe

- Append
  - passes all records in the pipe
  - executes the stage specified as a parameter
  - passes the output of that stage into the pipe
  - Example:
    - < filea list a | append < fileb list a | > filec list a
- Preface
  - executes the stage specified as a parameter
  - passes the output of that stage into the pipe
  - passes all input records
  - Example:
    - < fileb list a | preface < filea list a | > filec list a

Pipelines in REXX Execs

- Pipeline specification is a single REXX string
  - enclosed in quotes
  - can be continued by ending line with a comma

```
/* */
'pipe ',
  '< input file a |',    /* read file         */
  '> output file a'      /* writes output file*/
```

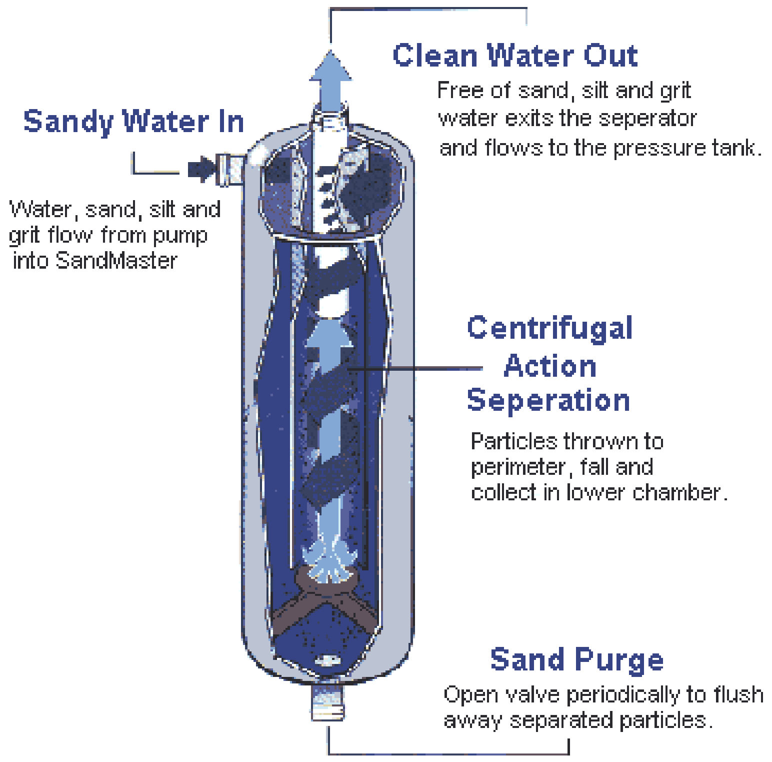
  - REXX variables may be inserted in the pipeline specification

```
/* */
parse arg fn
'pipe',
  '< ' fn ' oldfile a |',
  '> ' fn ' newfile a'
```

Exercise #3

- Write a pipeline to display your PROFILE EXEC on the console
- Write a pipeline to echo console input back to the console
- Write a REXX exec that can be issued from FILELIST that copies the specified file to your A disk with a filename of TEMP (keeping the existing file type)
- Modify the EXEC to copy a the file to the console as well as the A disk

A Sand Filter


Clean Water Out
Free of sand, silt and grit water exits the seperator and flows to the pressure tank.
Sandy Water In
Water, sand, silt and grit flow from pump into SandMaster
Centrifugal Action Seperation
Particles thrown to perimeter, fall and collect in lower chamber.
Sand Purge
Open valve periodically to flush away separated particles.

Selection Stages

- Selectively allow records to pass through to the next stage
  - **`find aaaa`** - passes records that begin with aaaa
  - **`nfind aaaa`** - passes records that don't
  - **`locate /aaaa/`** - passes records that contain aaaa
  - **`nlocate /aaaa/`** - passes records that don't
  - **`locate n`** - passes records that are at least n characters
  - **`between /aaaa/ /bbbb/`**
  - **`outside /aaaa/ /bbbb/`**
  - **`take n`** - passes only the first (or last) n records
  - **`drop n`** - passes all but the first (or last) n records

Exercise #4

- The file LABMSTR LIST B contains information about the sessions this week.
- Write pipes to read this file and then
  - List all the conference speakers
  - List all the session titles containing the string Pipe
  - List all session titles containing the string VM/ESA
  - Write a new file that contains only the Session, Title and Speaker records for each session

Transformation Stages

- Stages that alter the contents of records
  - `change /xxx/yyy/` - changes all occurrences of xxx to yyy
  - `strip` - removes leading and/or trailing blanks
  - `split` - splits records at a specified character
  - `join n /string/` - joins n records inserting string between them
  - `joincont` - joins records with or without continuation characters

Exercise #5

- Write pipes that read labmstr list b and then

A. Write a new file that contains the Session, Title, day and time records for each session
   - substitute actual days of the week for DAY0-DAY4
   - substitute actual times for AM1-3 and PM1-3
     - 8:00, 9:30,11:00, 1:30,3:00,4:30

B. Write a new file that lists the session numbers and titles as:
   - 21A -Intro to CMS Pipelines Lab

C. Write a new file that lists the session number, day and time as
   - 21A - Monday - 9:30
   - 27A - Tuesday - 8:00 - Tuesday - 8:00

The Spec Stage

- Builds output records from pieces of input records
- Modeled after the specs parameter of the copyfile command
  - Any number of pairs of input and output specifications
    - Input [transformation] output [alignment]

Specs Input Specifications

- Column Ranges

3-7	columns 3 through 7
3.5	5 columns starting in column 3
4;-2	column 4 through the second to last column
5-*	column 5 through the end of the record

- Word Ranges

w3-w7	words 3 through 7
w4;-w2	word 4 through the second to last word
w5-*	words 5 through the end of the record

- Field Ranges

fs % f3-f5	fields 3 through 5 (separated by %)

- Delimited Strings
  - /This string will go into the output record/

Specs Output Specifications

- Columns

5	output is to start in column 5
5.3	output is to occupy 3 columns starting in column 5
n	output is to be placed immediately after previous field
nw	output is to be placed one column after the end of the previous field

Other Useful Specs Keywords

- READ
  - used to create a singe output record from multiple input records
  - causes subsequent input requests to be satisfied by the next record
    - `specs 1-10 1 read 1-10 20`
- WRITE
  - used to create multiple output records from a single input record
  - passes the current output record and starts a new one
    - `specs 1-10 1 write 11-20 1 write 21-30 1`

Exercise 6

A. Rewrite Exercise 5B using specs column range input and output instead of join and change
B. Rewrite it again using word ranges instead of columns
C. Write a pipe to read labmstr list b and write one record per session:
   - Session 21A is given once
   - Session 26A is given twice
   - etc.

Pipeline Subroutines

- Can contain commonly used groups of stages to facilitate re-use
- Have a filetype of REXX
- Begin with the CALLPIPE statement instead of PIPE
- Input and output of the subroutine are indicated by the symbol *:

Pipeline Subroutine Example

- File SELECT1 REXX

```
/* */
  'callpipe',
    '*: |',               /* input comes in here */
    'outside /Abstract:/ /Eabstract:/ |',
    'nfind Time|',
    'nfind Speaker:|',
    'nfind Title:|',
    'nfind Room|',
    'nfind Day|',
    '*:'                  /* output goes out here */
```

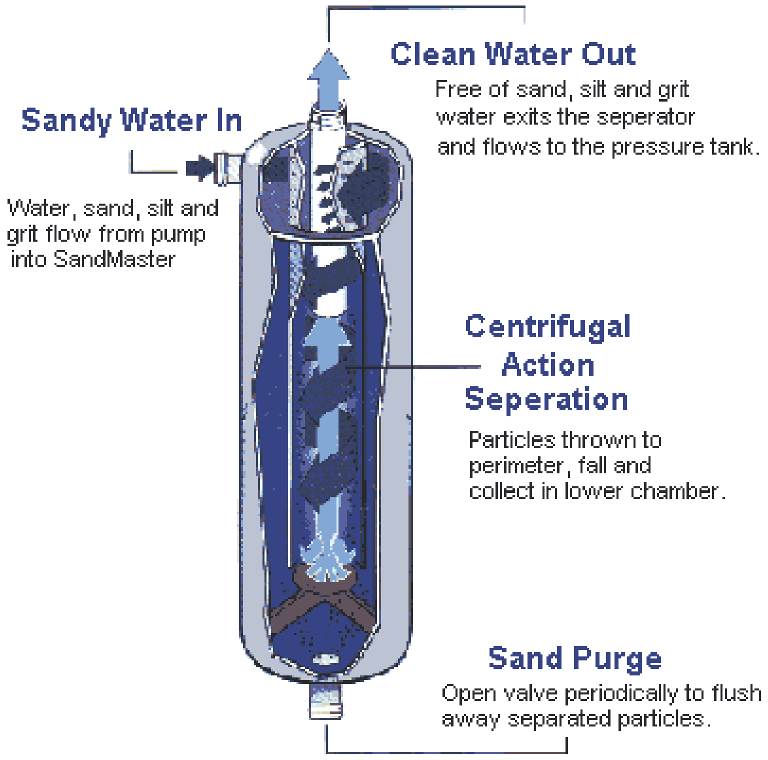
- Exercise 6B using subroutine select1

```
/* */
  'pipe',
    '< labmstr list a |',
    'select1 |',
    'specs w2 1 read /-/ nw w2-* nw |',
    '> temp list a'
```

Exercise 7

- Rewrite Exercises 5A and 5C to use a common pipeline subroutine to issue the change commands to convert the days and another pipeline subroutine to convert the times

Where Did the Sand Go?



Clean Water Out
Free of sand, silt and grit water exits the seperator and flows to the pressure tank.
Sandy Water In
Water, sand, silt and grit flow from pump into SandMaster
Centrifugal Action Seperation
Particles thrown to perimeter, fall and collect in lower chamber.
Sand Purge
Open valve periodically to flush away separated particles.

Multi-stream Pipes

- Selection stages send records not selected to a secondary output stream
  - identified by a label preceding the stage name
    - Labels are alpha numeric strings ending with a semi-colon
  - a label by itself followed by a stage separator indicates where the secondary output stage is to be connected
- Streams can be brought back together using the faninany stage
- The end character is necessary to indicate the end of a pipeline

A Simple Multi-stream Pipe

●

```
/* */
  'pipe (end \)',
    '< labmstr list a |', /* reads master file */
    'a1:find Day|',       /* get those that start with day*/
    'fixday|',
    'f1:faninany|',
    '> temp list a\',
    'a1:|',
    'fixtime|',
    'f1:'
```

A Complex Multi-Stream Pipe

Exercise #8

- Rewrite Exercise 6C to use multi-stream pipes
  - Select the necessary records
  - build a single output record for each session
  - Select those that are given once
    - modify those records to say ’is given once’
    - Modify the remaining records to say ’is given twice’

For More Information

- This week:
  - M26 - Pipelines Web CGI Techniques
  - M52 - Intermediate CMS Pipelines
  - M53 - Avoiding Pipelines Stalls
  - M54 - Advanced Pipelines Techniques
  - M55 - Lookup: The Plumber’s Swiss Army Knife
- On the Web:
  - The CMS Pipelines Page at Princeton University
    - http://pucc.princeton.edu/~pipelines
  - The IBM VM/ESA Pipelines Page
    - http://www.vm.ibm.com/pipelines/