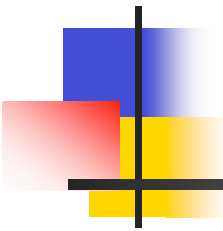


Now that I have Linux, What Do I Do with My JCL?



Replatform Technologies, LLC

May 21, 2007



Why listen to me?

- Moving applications since 1990
- Just finished project using these techniques
- Open system advocate
 - advocate solutions that do not tie you to a vendor

Bash?

Isn't there something better?

- JCL/JECL emulation environment
 - restrictive
 - proprietary
 - cost \$\$\$
- Automated conversions
 - Produce verbose code
 - cost \$\$\$
- Korn shell, Perl, and REXX
- Bash
 - Licensed under GNU public license
 - Free to download, probably already installed
 - Very easy to get started



What am I advocating?

- Modular programming.
- Not controversial.
- Needs to be advocated because some programmers are not doing it.



Sample converted JCL/JECL

```
#!/bin/bash
source stringent.sh || exit 1
source jcllibrary.sh
JclInit
JclPause "Make sure ONDISK.NAME.1 is available"
JclSetPrinter SYS001 "PrintJobName"
JclSetDD    SYS002 ONDISK.NAME.1 perm input
JclSetDD    SYS003 ONDISK.NAME.2 temp output
JclExecBatch TESTPROG
JclFtpSend  SYS003 user@server:dstname.txt
JclCommitPrint SYS001
JclFinalize
exit 0
```



Sample converted JCL/JECL

```
#!/bin/bash
source stringent.sh || exit 1
source jcllibrary.sh
JclInit
JclPause "Make sure ONDISK.NAME.1 is available"
JclSetPrinter SYS001 "PrintJobName"
JclSetDLBL SYS002 ONDISK.NAME.1 perm input
JclSetDLBL SYS003 ONDISK.NAME.2 temp output
JclExecBatch TESTPROG
JclFtpSend SYS003 user@server:dstname.txt
JclCommitPrint SYS001
JclFinalize
exit 0
```

PAUSE demo

```
localhost:~/jclbash pottmi$ vim myscript.sh
localhost:~/jclbash pottmi$ chmod 750 myscript.sh
localhost:~/jclbash pottmi$ ./myscript.sh
Press any key to continue:
localhost:~/jclbash pottmi$ cat myscript.sh
#!/bin/bash

read -s -n1 -p "Press any key to continue: "
echo # add a newline

localhost:~/jclbash pottmi$ █
```

convert PAUSE to library demo

```
localhost:~/jclbash pottmi$ cp myscript.sh jcllibrary.sh
localhost:~/jclbash pottmi$ vim jcllibrary.sh    # move code to function
localhost:~/jclbash pottmi$ vim myscript.sh     # call the function
localhost:~/jclbash pottmi$ ./myscript.sh
Press any key to continue:
localhost:~/jclbash pottmi$ cat jcllibrary.sh

function JclPause
{
    read -s -n1 -p "Press any key to continue: "
    echo # adds a newline
}

localhost:~/jclbash pottmi$ cat myscript.sh
#!/bin/bash

source ./jcllibrary.sh

JclPause

localhost:~/jclbash pottmi$ █
```



Giving the user choices

```
function JclFancyPause
{
    local Question=$1
    PS3=$Question

    select Answer in "Continue" \
                    "Exit with error" \
                    "Exit without error"
    do
        case $Answer
        in
            "Continue")           break           ;;
            "Exit with error")    exit 1        ;;
            "Exit without error") exit 0        ;;
            *)                    echo "Try again" ;;
        esac
    done
}
```

Fancy Pause demo

```
localhost:~/jclbash pottmi$ vim myscript.sh # change to Fancy Pause
localhost:~/jclbash pottmi$ ./myscript.sh
1) Continue
2) Exit with error
3) Exit without error
Do What?2
localhost:~/jclbash pottmi$ echo $?
1
localhost:~/jclbash pottmi$ ./myscript.sh
1) Continue
2) Exit with error
3) Exit without error
Do What?3
localhost:~/jclbash pottmi$ echo $?
0
localhost:~/jclbash pottmi$ cat myscript.sh
#!/bin/bash

source ./jcllibrary.sh

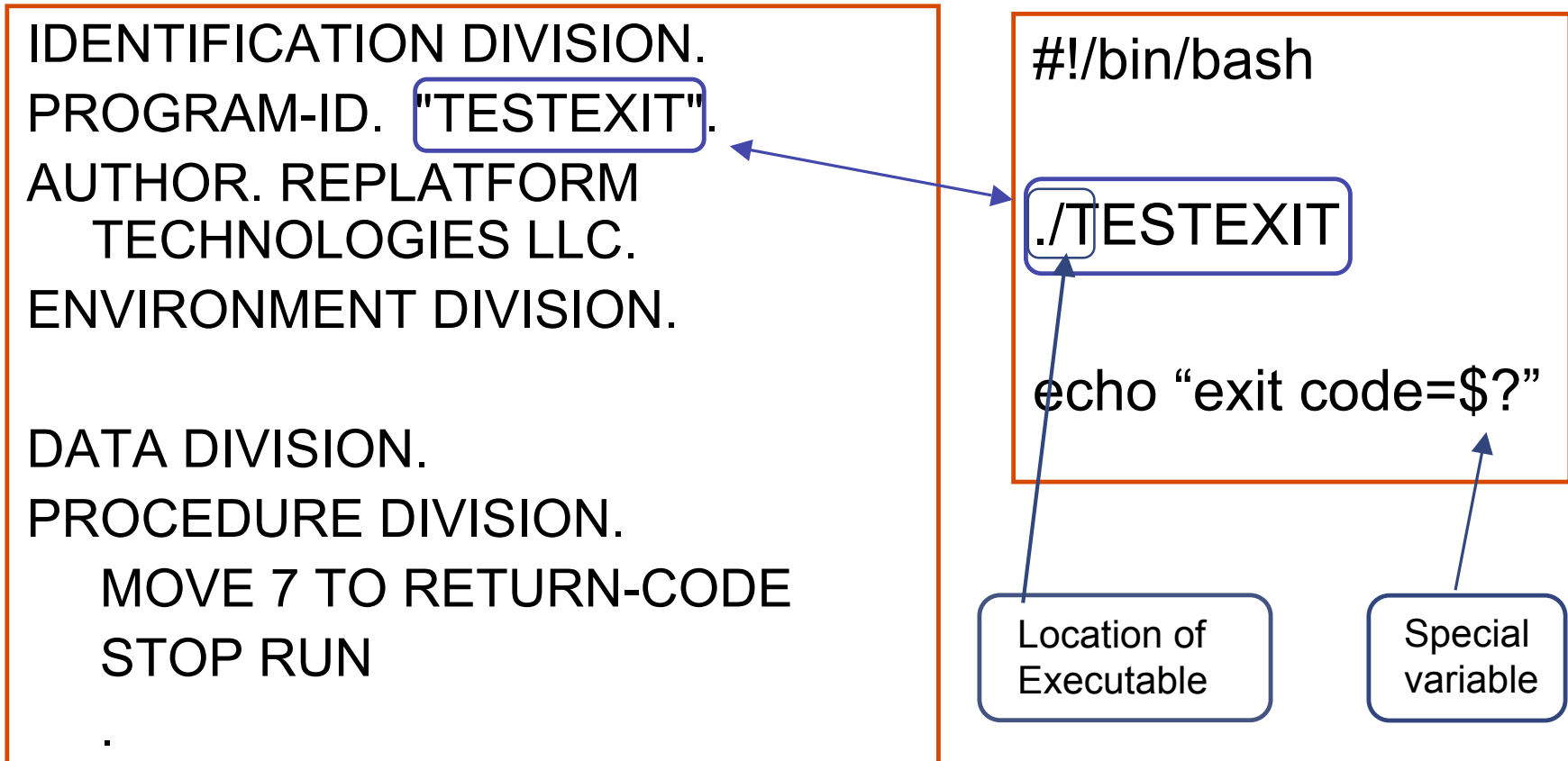
JclFancyPause "Do What?"

localhost:~/jclbash pottmi$ □
```

Exit codes

- Generally
 - Zero means success
 - Non-zero means failure
 - No authoritative standard (Several attempts)
 - My recommendation: use exit codes < 100
- Sample Exception: `cmp` (compares two files)
 - 0 means files are the same
 - 1 means files are different
 - 2 means failure such as could not open one of the files
 - Otherwise, failure such as memory fault
- Controlling
 - `bash` - `exit N`
 - COBOL - RETURN-CODE special register
 - C - `exit(N)`;
- Checking in `bash`
 - `$?` - special variable (see previous slide: `echo $?`)
 - `if (($? != 0))` to check for failure, rather than `if (($? == 1))`

Capturing the exit code from COBOL



Compile and run COBOL demo

```
$ cat TESTEXIT.cbl
  IDENTIFICATION DIVISION.
  PROGRAM-ID.  "TESTEXIT".
  AUTHOR. REPLATFORM TECHNOLOGIES LLC.

  ENVIRONMENT DIVISION.

  DATA DIVISION.

  PROCEDURE DIVISION.

      MOVE 7 TO RETURN-CODE
      STOP RUN
      .

$ cob2 -o TESTEXIT TESTEXIT.cbl
PP 5724-H44 IBM COBOL for AIX 2.0.0 in progress ...
End of compilation 1, program TESTEXIT, no statements flagged.
$ ./TESTEXIT
$ echo $?
7
$ echo $?
0
$ █
```



JclExecBatch function

```
function JclExecBatch
{
  local ProgName=$1

  ./$ProgName
  local ProgExitCode=$?

  if (( $ProgExitCode < 8 ))
  then
    JclFancyPause "$ProgName exited with $ProgExitCode, Do What? "
  elif (( $ProgExitCode != 0 ))
  then
    echo "ERROR:$ProgName exited with $ProgExitCode"
    exit $ProgExitCode
  fi
}
```

Download the full version of this
function from replatformtech.com

Run COBOL with library demo

```
$ cat myscript.sh
#!/bin/bash

source ./jcllibrary.sh

JclExecBatch TESTEXIT

$ ./myscript.sh
1) Continue
2) Exit with error
3) Exit without error
TESTEXIT exited with 7, do what? 2
$ echo $?
1
$ █
```



Bash Perils

- Continues (Maybe warns) after command error.
- Continues (Maybe warns) after unable to load library.
- No error or even warning about the use of an unset variable.
- Only the last command in pipeline sets \$?.
- Location of error may be hard to find.



Bash Perils with stringent.sh library

- Continues (Maybe warns) after command error.
 - Exits with Error Message.
- Continues (Maybe warns) after unable to load library.
 - Exits with Error message.
- No error or even warning about the use of an unset variable.
 - Exits with Error message including name of variable.
- Only the last command in pipeline sets \$?.
 - Makes all commands in pipeline subject to error checking.
- Location of error may be hard to find.
 - Error Messages include line number and library name



stringent.sh cookbook

- Download stringent.sh from www.replatformtech.com
 - Follow Download tab
- Make sure you are using a new version of bash.
 - `bash ./stringent.sh`, you will get an error if your bash version is old
- Make you scripts look like this:
 - `#!/bin/bash`
 - `source ./stringent.sh || exit 1`
- Run your scripts to catch errors.
- Use `errexitoff/on` to allow legitimate non-zero return.
- Use `traperr` to output error messages.

Run COBOL with stringent demo

```
bash-3.1$ vim myscript.sh
bash-3.1$ ./myscript.sh
ERROR: JclExecBatch(./jcllibrary.sh:~28)
      main(./myscript.sh:~6)
bash-3.1$ vim jcllibrary.sh
bash-3.1$ ./myscript.sh
1) Continue
2) Exit with error
3) Exit without error
TESTEXIT exited with 7, Do What? 2
bash-3.1$ echo $?
1
bash-3.1$ █
```

Force error demo

```
bash-3.1$ vim myscript.sh # remove stringent and induce error
bash-3.1$ ./myscript.sh
./jcllibrary.sh: line 31: errexitoff: command not found
./jcllibrary.sh: line 32: ./: is a directory
./jcllibrary.sh: line 34: errexiton: command not found
ERROR: exited with 126
bash-3.1$ vim myscript.sh # put stringent back in
bash-3.1$ ./myscript.sh
./jcllibrary.sh: line 29: $1: unbound variable
bash-3.1$ vim jcllibrary.sh
bash-3.1$ ./myscript.sh
ERROR: JclExecBatch(./jcllibrary.sh:~31) USAGE: JclExecBatch ProgName
        main(./myscript.sh:~6)
bash-3.1$ █
```



What you need to know about environment variables

- `$1, $2, ..., $9`
 - Positional parameters passed to function (or script)
- `WorkDir=/tmp/workdir`
 - Global variable with scope of script
- `local FileName=$1`
 - Local variable with scope of function and called functions
- `export SYS002=/mypath/myfile`
 - Global variable of scope of script (read and write) and called programs (readonly)

Using Environment variables as DLBLs and DD Names

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT afile ASSIGN TO SYS002
ORGANIZATION IS SEQUENTIAL
ACCESS IS SEQUENTIAL

DATA DIVISION.
FILE SECTION.

FD afile
RECORD VARYING IN SIZE 1 TO 4
DEPENDING ON a-length
DATA RECORD IS a-file

01 a-file.
05 a-var PIC 9(4).

MicroFocus

```
export dd_SYS002=/mypath/myfile.seq
```

IBM COBOL for AIX (VSAM)

```
export SYS002=VSAM-  
/mypath/myfile.seq
```

IBM COBOL for AIX (STL)

```
export SYS002=STL-/mypath/myfile.seq
```

DLBL / DD Name demo

```
$ export SYS002=VSAM-MIKEVSAM
```

```
$ ./dump256
```

```
$ od -t x1 MIKEVSAM |head -2
```

```
0000000 00 00 00 0a 14 4a 30 30 30 30 00 00 00 0a 14 4a  
0000020 30 30 30 31 00 00 00 0a 14 4a 30 30 30 32 00 00
```

```
$ export SYS002=STL-MIKESTL
```

```
$ ./dump256
```

```
$ od -t x1 MIKESTL |head -7
```

```
0000000 00 00 00 50 00 00 00 00 53 54 4c 53 01 02 03 04  
0000020 00 00 00 01 00 00 00 03 00 00 00 00 00 00 00 00  
0000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000060 00 00 00 00 10 00 00 00 00 00 00 01 00 00 00 04  
0000100 00 00 00 04 00 00 00 00 00 00 00 01 00 00 00 00  
0000120 00 00 00 04 30 30 30 30 00 00 00 04 00 00 00 04  
0000140 30 30 30 31 00 00 00 04 00 00 00 04 30 30 30 32
```

```
$
```



JclSetDLBL

```
function JclSetDLBL
{
  local DLBL=$1
  local LongName=$2

  local FileName=$LongName.seq

  # export dd_$DLBL=$FileName      # for microfocus
  export $DLBL=STL-$FileName      # for ibm stl
  # export $DLBL=VSAM-$FileName    # for ibm vsam
}
```

example: JclSetDLBL SYS002 ACCOUNT.REPORT

JclSetDLBL

```
function JclSetDLBL
{
  local DLBL=$1
  local LongName=$2
  local Location=$(getLocationPath $3)

  local FileName=$Location/$LongName.seq

  # export dd_$DLBL=$FileName
  export $DLBL=STL-$FileName
  # export $DLBL=VSAM-$FileName
}
```

example: JclSetDLBL SYS002 A.REPORT temp

```
function getLocationPath
{
  local Location=$1

  case $Location
  in
  temp)
    echo "/tmp/$$"
    ;;
  perm)
    echo "/some/path"
    ;;
  *) return 1 ;;
  esac

  return 0
}
```



Manipulating Data

	DFSORT	Unix	MortSort ¹
sort	SORT	sort	mortsort
reformat	INREC	cut	mortcut
merge	MERGE	-	mortmerge
pattern match	INCLUDE	grep	mortgrep
splitting	OUTFIL	split	mortsplit
common ²	-	comm	mortcomm

² splits two files into three files: records only in #1, only in #2, and common

¹ Available free but with some restrictions from Replatform Technologies LLC



simple sort

```
$ cat sample.txt  
occurs three times  
occurs once  
occurs twice  
occurs three times  
occurs twice  
occurs three times
```

```
$ sort sample.txt  
occurs once  
occurs three times  
occurs three times  
occurs three times  
occurs twice  
occurs twice
```



sort and piping gotchas

```
$ sort sample.txt |uniq -c  
1 occurs once  
3 occurs three times  
2 occurs twice
```

- Make sure set -o pipefail is on.
- Without it, bash will only detect failure of uniq.
- stringent.sh turns on this option.

```
$ sort sample.txt |uniq -c |sort -n -t' ' -k1  
1 occurs once  
2 occurs twice  
3 occurs three times
```

- Numeric sorts require additional options.
- Important because it does not always fail



Plethora of printing pitfalls

- Control Characters and Nulls not necessarily tolerated by UNIX.
 - Look into the 'tr' 'translate characters' UNIX command.
- First Column is print control on mainframe.
 - Look into 'cut' command to remove first character from every line
- Repeating a line to emphasize print.
 - Look into 'sed' 'stream editor' to remove the line.
- Printer channels used by COBOL program
 - Set export COBLPFORM=n:n:n:n:n:n:n:n:n:n (MicroFocus Only)



printing library functions

```
function JclSetPrinter
{
  # See JclSetDLBL:
  local DLBL=$1
  local FileName=/tmp/$$/pj/$2
  export $DLBL=$FileName
}
```

JclSetPrinter SYS002 theReport

FILE-CONTROL.

```
SELECT afile
  ASSIGN TO sys002
  ORGANIZATION IS
  LINE SEQUENTIAL
```

DATA DIVISION.

FILE SECTION.

```
FD afile
  RECORD CONTAINS 133
  DATA RECORD IS a-file
```

01 a-file.

```
05 a-var PIC X(133).
```



printing library functions

```
function JclCommitPrint
{
  local DLBL=$1
  local FileName
  eval FileName=\$$DLBL # DLBL contains name of var containing filename
  local TmpName=$FileName.tmp
  local PsName=$FileName.ps
  local PdfName=$FileName.pdf

  tr '\000' ' ' <$FileName \ # convert null to space
  | tr -C '[:print:\012\014\015]' '?' >$TmpName # convert control to ?

  psf -w -l60 -c133 $TmpName $PsName # convert text to postscript
  ps2pdf $PsName $PdfName # convert postscript to pdf

  lp $PsName # print the postscript
  cp $PdfName /archive # archive the pdf
}
```



UPSI Switches

If you have this in your COBOL programs:
`special-names.`

```
    UPSI-0 on status is upsi0-on
```

```
    UPSI-2 on status is upsi2-on.
```

```
if upsi2-on
```

then use this in your scripts:

```
export COBRTOPT="UPSI(10100000)"
```



Setting UPSI Switches Safely

```
function JclSetUPSI
{
    local UPSI=$1

    validateUPSI $UPSI

    if [[ -z ${COBRTOPT:-""} ]]
    then
        # runtime options are not already set
        COBRTOPT="UPSI($UPSI)"
    else
        # runtime options are already set
        if [[ $COBRTOPT == *UPSI* ]]
        then
            traperr "UPSI already set"; exit 1
        fi
        COBRTOPT="${COBRTOPT},UPSI($UPSI)"
    fi
    export COBRTOPT
}
```

```
function validateUPSI
{
    local UPSI=$1

    if [[ $UPSI =~
        "^[01][01][01][01][01][01][01][01]$" ]]
    then
        : # UPSI is 8 zeros or ones.
    elif (( $? == 2 ))
    then
        traperr "Malformed Regular Expression";
        exit 1
    else
        traperr "UPSI not 8 zeros or ones"; exit 1
    fi
}
```



Fragile Unix Commands

- `cp $srcfile $dstfile`
 - broken if \$srcfile has a space
- `cp "$srcfile" "$dstfile"`
 - broken if srcfile begins with -
- `cp -- "$srcfile" "$dstfile"`
- Same rules for:
 - `rm` - remove
 - `mv` - move or rename



Downloads

- psf by Tony Field
 - <http://ftp4.de.freesbie.org/pub/misc/plp/contrib/>
- ps2pdf
 - <http://www.cs.wisc.edu/~ghost/doc/cvs/Ps2pdf.htm>
- MortSort file sorting tools
 - <http://replatformtech.com>
- This presentation and other presentations
 - <http://replatformtech.com>
- stringent.sh & jcllibrary.sh
 - <http://replatformtech.com>



Things to think about...

- Job Steps
- Rerun Mode
- Test Mode
- Printer Channels
- Source Code Control



Contact Information

Replatform Technologies, LLC

1 877 247 6887

Info@replatformtech.com