z/OS Unix for all

Vit Gottwald, CA Technologies

presented by

Jan Samohýl, CA Technologies

August 9, 2011
Session 10067
Motivation for using z/OS UNIX

If you know Linux / UNIX
• Get started quickly
• Use familiar set of tools

If you know MVS
• Many programming tools
• Text processing utilities
• Have to support it …

Typical roadblocks

• EBCDIC
• Not a GNU system
• “Weird” error messages
• External security

• EBCDIC, but not 037
• Files instead of data sets
• “Shell” instead of ISPF
• How does it map to MVS?
Where to start

- IBM doc is great.
  - However, it takes a while to read through it. And they expect you to already have some basic knowledge of UNIX

- Tutorials on the web.
  - Is there no z/OS UNIX tutorial?

- Let’s pick a tutorial and comment on z/OS UNIX specifics
  - The rest of the session is loosely modeled after and to extend http://www.ee.surrey.ac.uk/Teaching/Unix/

No z/OS UNIX tutorial? Well, we haven’t seen one which is why we prepared this session.
How to start if you know nothing about UNIX

- Download a portable Linux distribution (e.g. Debian Live or Slax) and run it in a virtual machine (e.g. VirtualBox) on your PC (all for free)
- First try new commands in the Linux virtual machine where it is very likely going to work as expected
- Then try it on z/OS UNIX and compare results. If there is a difference check
  - This presentation for tips we found and included for you
  - The z/OS UNIX man pages or the IBM Command Reference
  - Other related IBM manuals

1) The z/OS UNIX Command Reference SA22-7802 is an equivalent of UNIX manual pages. It is basically a concatenation of all the man pages into a single document plus the description of related TSO commands.

1) See "IBM z/OS UNIX manuals" slide at the end of this presentation
UNIX Introduction

- [http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html)
UNIX Introduction

- Kernel
  - The heart of the system – provides UNIX services to it’s callers (“system calls”)
  - In z/OS part of the Basic Control Program (BCP)
- Shell
  - An interface between a User and the Kernel
  - Accepts, interprets, and executes your commands
- File system
  - Hierarchical directory structure for storing data (in “files”)
  - A whole file system in z/OS UNIX could be a single Data Set (HFS or zFS)

http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html

could be – but typically is not, multiple datasets are used to have a single dataset for each logical part of the file system, see "Planing" manual or attend Session 9875: "z/OS Basics: The z/OS UNIX Shared File System Environment and How It Works" by Jim Showalter

Another essential part of the UNIX environment is formed by application programs. Especially the standard set of programs available on every UNIX, often called “commands” or “utilities”.

From now on if appropriate there will be a link to the relevant part of the web tutorial at the bottom of each page.
UNIX terminal modes:
   a) Line at a time = line mode = canonical mode
   b) Process every keystroke immediately = raw mode = non-canonical mode

3270 = screen at a time

In 3270 or vt100/xterm in line mode you lose many of the shell productivity features. Those could be a real difference between a pleasant and a mundane session and whether you develop a passion or a hatred for the system.
The de facto standard application for vt100/xterm emulation over telnet or ssh on the MS Windows platform is Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/).

The UNIX terminal operates in one of two modes

1) Line / canonical – which sends line at a time to and from the UNIX box, each character received is immediately processed

2) Character / non-canonical / raw – every key pressed is sent to the UNIX box for processing and echoed back on the screen (unless echo is turned off)

Line mode is typically convenient if you are working on a system with response over 150 ms so that you don’t have to wait for each individual keystroke to be processed before you can type the next one. You lose a lot of interactive features of the shell and other applications though (e.g. completions).
To start the session
    enter “OMVS” from TSO command line or “TSO OMVS” from ISPF command line.
To quit the session:
    a) type "exit" and hit ENTER
    b) hit PF2 (SubCmd), then type in "quit" and hit ENTER

Are you curious what the ESC=¢ stands for?

To simulate Unix terminal's control sequences (like Ctrl-D) copy the character specified in ESC=¢ on the screen followed by the character (for Ctrl-D it is the 'D' character). In this case you want to send a Ctrl-D sequence, you will type into the command line "cd" and then hit ENTER to send it for processing. This particular sequence will log you out of the session.

More about TSO OMVS in session 10065 - TSO OMVS and You: What to Make of UNIX System Services
on Thursday, August 11, 2011: 11:00 AM-12:00 PM
Navigating the directory structure

- [http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html)
Navigating the directory structure

- Described in every tutorial (including the one we picked)
- Concept of a “current working directory” or “.”
- Concept of a “parent directory” or “..”
- Concept of a “home directory” or “~”

- Basic commands
  - `pwd` print current working directory
  - `mkdir` create directory
  - `cd` change (current working) directory
  - `ls` list files in (current working) directory

http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html
Type ISHELL in your TSO session or TSO ISHELL from ISPF to invoke the ISPF Shell
Type a command next to a directory or a file to perform an action. A question mark will give you a menu of available actions. This slides shows available actions for a directory.
This slides shows available actions for a file.
If you want to execute unix commands from ISHELL, type "sh command" or "ex command".

In the first case you will run your command within your shell {sh | tcsh | bash}. That means that /etc/profile and a profile in your home directory relevant to the particular shell will be processed before running the command.

In the latter case, the command will be run directly without any interaction with shell.

In both cases you will get the output in ISPF browse like panel. You can scroll, search, etc.

find /etc/ will list all files in your /etc directory.
Shells can be used both for interactive work and batch processing.

z/OS shell comes from the Bourne Shell family. Unfortunately it is not user friendly over a telnet/ssh session.

On the other hand tcsh has been designed for interactive work and fits well to an interactive telnet/ssh session.

BASH (Bourne Again Shell) is a port of GNU BASH, a modern shell used in most Linux distributions. The z/OS UNIX port is not complete and not supported by IBM, but generally works as expected.
Manipulating files and directories

- [http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html)
Manipulating files and directories

- Manipulating files/directories
  - cp – copy files/directories
  - mv – move files/directories
  - rm, rmdir – remove files, directories
- Displaying contents of a file on the screen
  - head – print couple lines from the beginning of a file
  - tail – print couple lines from the end of a file
  - cat – print all the contents of the file on screen sequentially
  - less – print as much as fits on one screen, let the user scroll

http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html
Manipulating files and directories

- Searching the content of files
  - less – so called “pager” - allows scrolling (both forward and backward), searching for a text pattern
  - grep – displays lines that match a “regular expression”
  - wc – ‘Word Count’ – counts characters, words, lines

- ISHELL
  - You can browse, view, and edit files using the ISPF editor

http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html
Unix files are just a stream of bytes without an internal structure. It is the program who interprets the contents, not the file system. The picture shows a dump of a file called "test" produced by an "od" (Octal Dump) command. The "\n" symbol is the standard way of referring to the newline character.
The picture on this slide shows a (text) copy of the file test into a dataset with fixed record length. Notice that newline characters have been removed and replaced with blank (X'40') characters padding each line to its full length.

Also notice that even though the bytes X'5F', X'AD', X'BD', X'B0', X'BA', X'BB' haven't physically changed they are displayed as different characters. This is due to the setting of the terminal to display data as IBM-037 on this (3270) terminal, while to IBM-1047 in the unix terminal on previous slide.
The extra quotes around "hlq.desti(nation)" are present because they tell the shell this is a single parameter to be passed to the program 'cp'. The shell removes them before passing the argument to the program.

Copying "program objects" from a file system to a PDS can result in unpredictable results since the conversion to a "load module" may or may not be successful. Use PDS/E instead.

The picture on this slide shows the result of copying the very same file "test" into a dataset having a fixed record length but this time in a binary mode via cp with the -B option. This time we got a single record including X'15' (newline) characters and the record is padded with binary zeros to its full length.

If truncation is to occur (record length smaller than a line length) cp fails not copying anything and issuing an error message.
Copying data between files and datasets

- To convert files from one encoding to another you can use the `iconv` command

- Sample usage

```bash
iconv -f IBM-037 -t IBM-1047 < infile > outfile
```

this will convert file `infile` in code page 037 to code page 1047 and save the output in file `outfile`

http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html

In case you are interested in the history of IBM 1047, this is an interesting discussion http://groups.google.com/group/bit.listserv.ibm-main/msg/0b633f8476306da0 it seems to relate to SHARE.
Copying data between files and datasets the 'batch' way

- **OCOPY**
  - Copy and optionally convert between IBM-037 and IBM-1047
  - A TSO command, copies between two allocated DD names
  - Sample

```
//COPYSTEP EXEC PGM=IKJEFT01
//UNIX DD PATH='/tmp/a'
//MVS DD DISP=SHR,DSN=GOTVI01.JCL(A)
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
OCOPY INDD(UNIX) OUTDD(MVS) TEXT
/*
```

There is also a BPXCOPY utility. However I never used it.
Copying data between files and datasets the MVS way

- There is a whole set of TSO commands
  - **OPUT** - copy data set [member] into a file
  - **OGET** – copy file into a data set [member]
  - **OPUTX** - copy members from a PDS(/E) to a directory
  - **OGETX** - copy files from a directory to an PDS(/E)

[http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html)

See the manual (click on the particular command) to find out more about these and programs.
Input/Output redirection, pipes


http://www.ee.surrey.ac.uk/Teaching/Unix/unix3.html
When executing commands in z/OS UNIX shell, we do not have a JCL available to specify the input and output as we do in batch. To provide for this, each command being executed has a predefined "standard input", "standard output", and "standard error" to read input and write output and errors. Standard input is typically mapped to the input from keyboard, standard output and standard error to the terminal screen.

Special shell syntax makes it possible to redefine the standard input/output/error and let the commands to read/output whatever the user asked for.
Several MVS specific utilities are included. If you want to list PDS members from a shell, there is a ‘pdsdir’ utility. If you want to pipe from/to data set there are ‘readmvs’ / ‘writemvs’ utilities.

There are also many useful tools including BASH and several GNU utilities, most notably ‘GNU grep’ which also supports processing of datasets (even groups of datasets by specifying a recursive mode via –r and providing a generic DSN pattern like in ISPF ‘Data Set List Utility’ (i.e. in the form that Catalog Search Interface understands).
Filesystem security (access rights)

- [http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html)
The `ls` command also has the `-E` switch that prints all that `-l` does, but provides some more z/OS UNIX specific attributes like APF authorization or if a program library is shared or not.
Files for which the output of `ls -l` starts with an "l" are so called "symbolic links"

In z/OS UNIX there is a special type of a symbolic link whose name starts with a `$SYSNAME/...` This refers to the symbolic `&SYSNAME` as defined in SYS1.PARMLIB for a SYSPLEX environment. If SYSPLEX is specified as NO in SYS1.PARMLIB(BPXPRMxx), then `$SYSNAME` is replaced with `/SYSTEM` when the symbolic link is resolved.

For more see
From ISPF or OMVS we can use OBROWSE (or ISPF P.3.2) to show output similar to `ls -l`
Filesystem security

- Unlike on other UNIX platforms, in z/OS UNIX you have to use an external security product
- OMVS segment
  - Part of user security profile
  - Has to be defined to a user in order to use z/OS UNIX
  - A default can be provided for users who do not have their own
  - Specifies your User ID (number), login shell, home directory

- Comparing UNIX, MVS, and z/OS UNIX security
  - [http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix5.html)
### Comparing UNIX, MVS, and z/OS UNIX security

<table>
<thead>
<tr>
<th>Category</th>
<th>Traditional UNIX</th>
<th>MVS</th>
<th>z/OS UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>User identity</td>
<td>Users are assigned a unique UID, a 4-byte integer and user name.</td>
<td>Users are assigned a unique user ID of 1-to-8 characters.</td>
<td>Users are assigned a unique user ID with an associated UID.</td>
</tr>
<tr>
<td>Security identity</td>
<td>UID</td>
<td>User ID</td>
<td>UID for accessing traditional UNIX resources and the user ID for accessing traditional z/OS resources</td>
</tr>
</tbody>
</table>
### Comparing UNIX, MVS, and z/OS UNIX security

<table>
<thead>
<tr>
<th>Category</th>
<th>Traditional UNIX</th>
<th>MVS</th>
<th>z/OS UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login ID</td>
<td>Name used to locate a UID</td>
<td>Same as the user ID</td>
<td>Same as the user ID</td>
</tr>
<tr>
<td>Special user</td>
<td>Multiple user IDs can be assigned a UID of 0.</td>
<td>RACF® administrator assigns necessary authority to users.</td>
<td>Multiple user IDs can be assigned a UID of 0 or users can be permitted to BPX.SUPERUSER.</td>
</tr>
</tbody>
</table>
### Comparing UNIX, MVS, and z/OS UNIX security

<table>
<thead>
<tr>
<th>Category</th>
<th>Traditional UNIX</th>
<th>MVS</th>
<th>z/OS UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity change from regular user to superuser</td>
<td>The <code>su</code> shell command allows change if user provides password for the root. Password phrases are not used in traditional UNIX security.</td>
<td>No provision for unauthorized user to change identity.</td>
<td>The <code>su</code> shell command allows change if the user is permitted to BPX.SUPERUSER or if the user provides the password or password phrase of a user with a UID of 0.</td>
</tr>
<tr>
<td>Terminate user processes</td>
<td>Superuser can kill any process.</td>
<td>MVS operator can cancel any address space.</td>
<td>Superuser can kill any process.</td>
</tr>
</tbody>
</table>
Copying data over network

- Over network
  - FTP
  - Open SSH (SFTP, scp), part of IBM Ported Tools for z/OS
  - cURL – great utility originally for downloading and uploading data over HTTP, FTP, FILE and even more protocols, part of Supplementary Toolkit for z/OS
‘quote site ispfstats’ enables ISPF statistics processing when updating members of PDS or PDSE. It stores the date, time and user account name.

‘quote site sbdataconn=(ibm-037, iso8859-1)’ sets the conversion table for file transfer in ascii (text) mode. Beware that z/OS UNIX uses IBM-1047 as the default character encoding while MVS uses IBM-037. These two encodings differ only in a couple of characters, among which are the ^ [ ] symbols. If you have these characters make sure to set the right encoding.

‘quote site filetype=jes’ switches the ftp client to JES mode and allows you to submit jobs (put) and download the output of other jobs (get).

‘quote help site’ shows the available site options with short help.

For more information see z/OS Communications Server IP User’s Guide and Commands[http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/F1A1B980/5.64?SHELF=F1A1BKA1&DT=20080602140814]
We do not have enough room in this session to go into details. Instead we reference a session from SHARE in Anaheim as a good point to get started.

- **http://www.ee.surrey.ac.uk/Teaching/Unix/unix7.html**

- **C: The “Dark Side” of System z?** by Brandon Tweed
  - Hosted by z/NextGen at SHARE in Anaheim
  - Introductory session to software development in C on z/OS

http://www.ee.surrey.ac.uk/Teaching/Unix/unix7.html
Environment / SHELL Variables

- [http://www.ee.surrey.ac.uk/Teaching/Unix/unix8.html](http://www.ee.surrey.ac.uk/Teaching/Unix/unix8.html)
Environment / SHELL Variables

- A way of passing information from SHELL to the commands it executes
- Examples
  - PATH – directory names separated by a colon – where the shell looks for commands to execute
  - HOME – your home directory
  - USER – your user name under which you logged on
  - HOST – name of the system you are logged on
- Printing the current values
  - echo $variable_name
  - e.g. "echo $PATH", "echo $HOST", "echo $HOME", etc.
  - "env" command prints all currently defined variables

http://www.ee.surrey.ac.uk/Teaching/Unix/unix8.html
If you are an application programmer you should understand what parameter list your program receives when run it from a shell, from a JCL or from a TSO. Each of these environments, provides a different convention.

In case you are interested to see the difference in parameter passing, see:

Standard JCL PARM passing:

*  

z/OS UNIX exec() parameter and environment passing:

*  
http://publib.boulder.ibm.com/infocenter/zos/v1r12/topic/com.ibm.zos.r12.bpxb100/exc.htm#exc

TSO Command Processor Parameter List (CPPL):

*  
Web references

1) Overview of the UNIX* Operating System
   http://www.bell-labs.com/history/unix/tutorial.html

2) UNIX Tutorial for Beginners
   http://www.ee.surrey.ac.uk/Teaching/Unix/

3) UNIX TOOLBOX – a pool of typical usage examples
   http://cb.vu/unixtoolbox.xhtml

4) The Grymoire - home for UNIX wizards – sed, awk, etc.
   http://www.grymoire.com/Unix/
Legacy UNIX books - a treasure-trove

At times things do not make sense. Sometimes they got lost in translation and it is helps to go see what they were originally created for

1) The UNIX Programming Environment, Brian W. Kernighan, Rob Pike, 1984


I recommend reading 1). Things start to make sense when explained by original authors. First 170 pages are suitable for everyone learning the subject. The second half is for programmers. It has a great index at the end of the book!

Consider the book 1) a guide and 2) a reference manual.
Further resources

- [UNIX for the Mainframer](#) by David Horvath. ISBN 0-13-632837-7

MVS-OE stands for MVS Open Edition, the original name of the product later called Unix System Services and nowadays z/OS UNIX.
IBM z/OS UNIX manuals

- z/OS 1.12 information center
  - SA22-7802-13 Command Reference
  - SA22-7807-11 Messages and Codes
  - GA22-7800-18 Planning
  - SA22-7805-08 Programming Tools
  - SA22-7801-13 User's Guide

- ... and more at ...
Thank you for your attention!

Please do not forget to fill in evaluation forms.

Session 10067
Additional material about running z/OS UNIX commands within batch is appended at the end of the session slides. It is included in proceedings.
The Next Generation of Mainframe Professionals.

Additional Material Follows
Running z/OS UNIX tools in batch
Running UNIX tools in batch

- IBM provides a utility which can be used to run UNIX commands from batch
- The utility has two entry points
  - BPX BATCH
    - Does a “fork” that creates a new address space for the commands to run, hence they lose access to datasets defined in the job step
  - BPX BATSL
    - Does a “local spawn”, runs the commands in its address space
- For documentation see
  - z/OS V1R10.0 UNIX System Services Command Reference
- Also see Dovetail’s [Co:Z Batch](#) for more options

See also www.dovetail.com an their COZBATCH utility, it's free and has some advantages over BPXBATSL.
Running in batch, cont’d

- Sample step that waits for 10 seconds
  //       SET WAIT='10'
  //SLEEP EXEC PGM=BPXBATCH,PARM='SH /bin/sleep &WAIT.'

- Sample step executing a Bash script
  //BASH EXEC PGM=BPXBATCH
  //STDIN DD DUMMY
  //STDOUT DD SYSOUT=*
  //STDERR DD SYSOUT=*
  //STDPARM DD *
  SH /u/gotvi01/script/diff.sh
  """/SYS1.PROCLIB(HLASMCL)"
  """/SYS1.PROCLIB(HLASMCLG)"
  /*
Sample script \

```bash
#!/sys/s390util/bin/bash
# Work files
TMP1="/tmp/$$.1.tmp"
TMP2="/tmp/$$.2.tmp"
# Ignore record numbers
cut -c 1-72 $1 > $TMP1
cut -c 1-72 $2 > $TMP2
# Compare the content
/sys/s390util/bin/diff -auw \
$TMP1 $TMP2 \
--label $1 --label $2
# Delete work files
rm $TMP1 $TMP2
```

Sample output

```
--- //SYSI.PROCLIB(HLASMCL)'
+++ //SYSI.PROCLIB(HLASMCL)'
@@ -1,9 +1,10 @@
-ASMASMCL PROC
+ASMASMCL
+ASMASMCL

//THIS PROCEDURE RUNS THE HI
//NEWLY ASSEMBLED PROGRAM AN
//THE LINK-EDIT IS ACCOMPLIS

//EXEC PGM=ASMAS MO,PAR
//STEPLIB DD DSN=ASA SSMC
-24.3 +25.4
+DCBSN1=I
DCBSN1=I
DCBSN1=I
DCBSN1=I
```

Sample script \\u/gotvi01/script/diff.sh – compare content of two JCL procedures with GNU diff

In most shells, $$ is a variable representing currently running process’ PID. It helps us generate random, unique file names for temporary files.

$1 is the value of first parameter passed during script invocation. $2, $3, … $n are the respective n-the parameter. $0 is the command string used to invoke the script.

Backslash ‘\’ operates as a line continuation character.

-auw tells GNU diff to

a) treat input as text files even when it encounters non-printable characters

u) use “universal diff” format for output (a de facto standard in open source community)

w) ignore all white spaces (space is equivalent to tab and so is any sequence of them)

beware that the port of GNU diff to z/OS UNIX suffers from some non-functional options (e.g. diff --help tells the user that --help is an unknown parameter and to type diff --help for help) however these are of low practical importance (help can be found on the GNU project documentation web page)
Accessing datasets

- UNIX is implemented in C
- Opening files in C
  - fopen() – part of ANSI C standard, IBM implementation allows use of special file names to open datasets and ddnames
    - //proclib – refers to DSN=tsopfx.PROCLIB
    - //sys1.maclib' – refers to DSN=SYS1.MACLIB
    - //dd:sysout – refers to sysout DDNAME
  - open() – a POSIX standard function, current implementation does not allow dataset processing
- When passing dataset names from shell, you have to enclose them in quotes like "//sys1.maclib'" not to lose the apostrophes in the dataset name

For more details regarding IBM XL C/C++ runtime library see

z/OS V1R10.0 XL C/C++ Programming Guide (2.9.1.1 Using fopen() or freopen())
Accessing datasets, cont’d

- Many UNIX commands use fopen() and are able to read/write sequential datasets (members of PDSs in particular)
  - cat – print the content of its input to output
  - cut - print only specified columns/fields from its input to output
  - cp – copy either sequential data or program objects
  - diff – compare content of two sequential text files
  - sed – stream editor – regular expressions based utility
  - pdsdir – utility for listing members of a PDS
    - Provided as part of z/OS UNIX Tools [http://www-03.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty2.html]
    - This is a REXX script, not a C program, argument is pure DSN

The information on this slide is solely derived from the author’s experimenting with various utilities and may be wrong. It is certainly NOT complete and serves only as a hint for you what to try.
Regular expressions
Regular expressions

- Provide a powerful text matching mechanism
- Basic tokens (characters, numbers, white spaces, start/end of line, word)
- Repetition specifiers (once, twice, at least once, any, … )
- Grouping mechanism - new tokens by combining the above

- Sample:
  - `'^'` – match lines beginning with a blank
  - `'^[^ ]'` - match lines starting with a non-space (e.g. a label)
  - `'\s*\s'` – match lines ending with a back-slash (that may or may not be followed by blanks)

The description here is overly simplified for the purpose of this presentation. There are many dialects of regular expressions.
For more information about IBM implementation see

For more information about regular expressions and their various dialects see the book “Mastering regular expressions” by Jeffrey E.F. Friedl, O’Reilly Media1997
Regular expressions cont’d

- A mechanism for specifying text patterns by a logical grouping rather than by column position
- Tools utilizing regular expressions (aka regexes)
  - `grep` – process a text input and print lines matching a regex
  - `sed` – process a text input substitute matching patterns with specified replace strings/patterns
  - vi(m) – text editor with support for regex match/substitute
  - Many modern programming languages and editors provide support for some dialect of regular expressions (perl, python, java, C#, …)
GNU grep

- Provided as part of tools ported to z/OS UNIX “Ported Tools”
- Supports catalog search (//h1q.mlq*)
- Following command searches SYS1.MACLIB for all lines containing an SVC 34 instruction according to the logic HLASM understands its source code

```
/sys/s390util/bin/grep -ir "^\([.]\)?[@#$a-z0-9]\+\\)? +svc +34' "/'sys1.maclib"
```
To display the line numbers where the particular match occurred use the -n option.
UNIX tools and their MVS analogons

- UNIX and MVS are build on completely different paradigms and comparing available tools is not straightforward
- The goal is to provide a starting point for those who know one of them and want to learn about the other

<table>
<thead>
<tr>
<th>UNIX</th>
<th>MVS</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp</td>
<td>IEBGENER, IEBCOPY</td>
</tr>
<tr>
<td>mv, rm</td>
<td>IDCAMS</td>
</tr>
<tr>
<td>diff</td>
<td>SuperC/SuperCE</td>
</tr>
<tr>
<td>grep</td>
<td>Search-For/Search-ForE</td>
</tr>
<tr>
<td>pax</td>
<td>TRSMAIN</td>
</tr>
<tr>
<td>shell scripts</td>
<td>REXX execs + ISPF services</td>
</tr>
<tr>
<td>find, locate</td>
<td>Catalog Search Interface + LISTDS</td>
</tr>
</tbody>
</table>

Documentation for

IEBGENER, IEBCOPY, IEBUPDTE

z/OS V1R10.0 DFSMSdfp Utilities
[http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/DGT2U140/CCONTENTS?DT=20080521090625]

SuperC/SuperCE, Search-For/SearchForE
z/OS V1R10.0 ISPF User's Guide Vol II

IDCAMS
z/OS V1R10.0 DFSMS AMS for Catalogs
[http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/DGT2I270/CCONTENTS?DT=20080602132758]
Running in batch II
Running in batch II

- Remember that C compiler produces LE compliant prgrms!
- Program objects can be copied between z/OS UNIX file system and PDSE libraries
  - Multiple utilities allow this: cp, OCOPY, IEBCOPY, the binder
  - The easiest way is to use: cp -X /bin/diff //lod
- You might then be able to run the program from the new location

  //DIFF EXEC PGM=DIFF,PARM='POSIX(ON) / -c dd:in1 dd:in2'
  //STEPLIB DD DISP=SHR, DSN=GOTVI01.LOD
  //IN1 DD DISP=SHR, DSN=CBC.SCCNPBC(CBCC)
  //IN2 DD DISP=SHR, DSN=CBC.SCCNPBC(CBCCL)

Language Environment options, separator, parameters passed to the program

For more information how to copy data and executables between z/OS UNIX file system and datasets see:

z/OS V1R10.0 UNIX System Services User's Guide, 21.0 Copying data between the z/OS UNIX file system and MVS data sets


For more information about Language Environment and passing parameters to LE compliant programs see:

z/OS V1R1.0 Lang Env Prog Guide, 1.5.1.3 Specifying Run-Time Options in the EXEC Statement

[http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/CEEA2100/1.5.1.3?SHELF=CEE2BK00&DT=20010116123540]

IBM XL C/C++ compiler produces LE compliant code by default. You can change the default behavior through EXECOPS / NOEXECOPS compiler options or #pragma runopts macros.

[http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ceea2190/1.9.2?DT=20080603042013]
Final tips

• When running BPXBATCH or BPXBATSL you sometimes get a message like

BPXM047I BPXBATCH FAILED BECAUSE SPAWN (BPX1SPN) OF DIFF FAILED WITH RETURN CODE 00000081 REASON CODE 053B006C

• To get a more detailed message, run `bpxmtext` UNIX command and pass it the reason code from the message

  > bpxmtext 053B006C
  BPXFSSTA 11/16/07
  JRFilNotThere: The requested file does not exist

  Action: The service cannot be performed unless the named file exists.
Final tips, cont’d

- Make sure there are no record numbers in positions 73-80 of input files processed by UNIX utilities. They do not handle these positions as special and usually fail.

- If you are on a SYSPLEX with shared JES make sure you use the right `/^JOBPARM S=system` parameter to run on the right LPAR (unless you have a shared file system).

- Also watch out for data encoding. MVS uses IBM-037 while z/OS UNIX uses IBM-1047. Most characters match, but `^` and some more do not. Use `iconv` or `OCOPY` to translate.