#SHAREorg

SHARE
Technology · Connections · Results

z/OS UNIX for all

Vit Gottwald
CA Technologies

August 8, 2012

This session was previously presented at SHARE in Atlanta,

March 15, 2012

SHARE in Anaheim 2012

- **Motivation**
- Introduction to UNIX
- Interfacing with z/OS UNIX
- Shells and scripts
- Copying between files and datasets
- ISHELL

Motivation for using z/OS UNIX

If you know Linux / UNIX

- Get started quickly
- Use familiar set of tools

If you know MVS

- New programming tools
- Text processing utilities
- Connecting to the web

Typical roadblocks

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

- EBCDIC, but not IBM-037
- Files instead of data sets
- “Shell” instead of ISPF
- How does it all map to good old MVS?

Where to start

- IBM doc is great
  - However, it takes a while to read through
  - You are expected to already have 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 extends http://www.ee.surrey.ac.uk/Teaching/Unix/

Where to start

- 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)
- Play with the Linux virtual machine
  - there are many tutorials on the web
  - don't be afraid to experiment
  - most of what you learn will be applicable on z/OS UNIX
- When something in z/OS UNIX doesn't work as expected
  - go back to your Linux box and compare results
  - check this session
  - Search man pages or IBM z/OS UNIX Command Reference.

- Motivation
- **Introduction to UNIX**
- Interfacing with z/OS UNIX
- Shells and scripts
- Copying between files and datasets
- ISHELL

Introduction to UNIX

- 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 is stored in one or more Data Sets (HFS or zFS)

Introduction to UNIX

- “commands” or “utilities”
  - standard set of programs available with every UNIX
  - described in the POSIX standard and its extensions http://en.wikipedia.org/wiki/Single_UNIX_Specification
  - Sample categories:
    - Administration tools
    - Text processing utilities
    - Programming tools
  - SHARE in Seattle Session 2285, Basic UNIX Shell Commands for the z/OS System Programmer (link)

- Motivation
- Introduction to UNIX
- **Interfacing with z/OS UNIX**
- Shells and scripts
- Copying between files and datasets
- ISHELL

Interfacing with z/OS UNIX

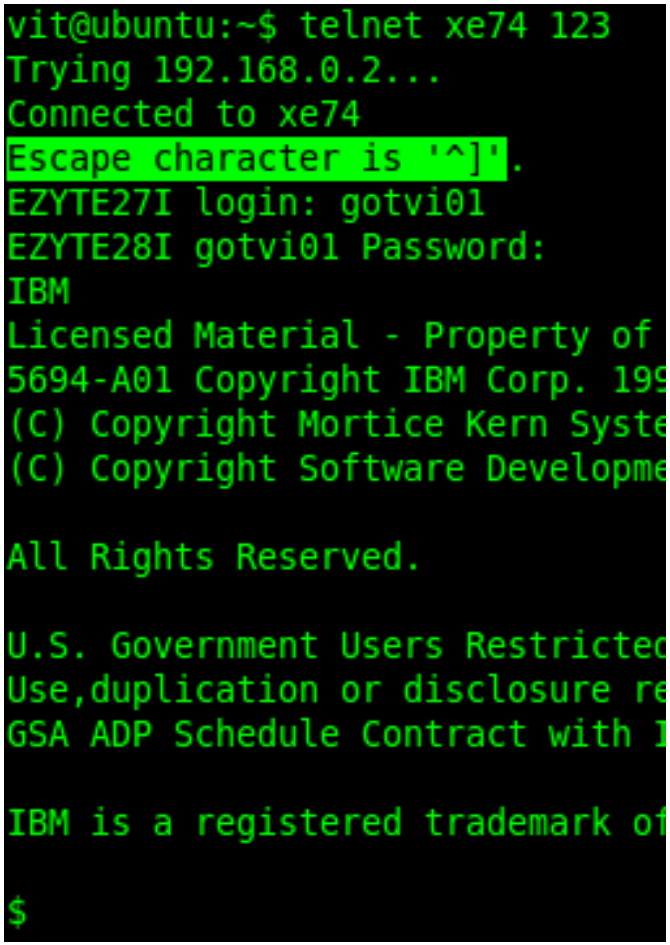
- UNIX terminal (VT100, VT220, xterm)
  - Interactive work via telnet or ssh
  - This is the standard (and typically only way) on other platforms
- 3270 (TSO OMVS, ISPF Shell)
  - OMVS
    - Type in a command or two and read the output
  - ISHELL
    - The MVS-like way of doing things (through ISPF panels)
- Batch
  - UNIX services (APIs) in application programs
    - TCP/IP, Java, Web servers, Application servers
  - UNIX tools to process datasets (text processing tools)

UNIX terminal over telnet

- Telnet
  - network protocol used on Internet or local area networks to provide a bidirectional interactive text-oriented communication
  - operates in one of two modes (known as "line discipline")
    - **Line / canonical** – sends characters to and from the UNIX box line at a time – when you type, echoing to your terminal and line editing takes place locally; the whole line is sent only after you press the ENTER key
    - **Character / non-canonical / raw** – every key pressed is sent immediately to the remote machine for processing; every key stroke has to be sent, remotely processed, and returned back for echoing on your screen

UNIX terminal over telnet

- From Linux or UNIX system you can login through `telnet ip_address port`
- Watch out for message "`Escape character is …`"
- '`^]`' stands for `Ctrl+]` and represents the "Escape character" or "Escape key"

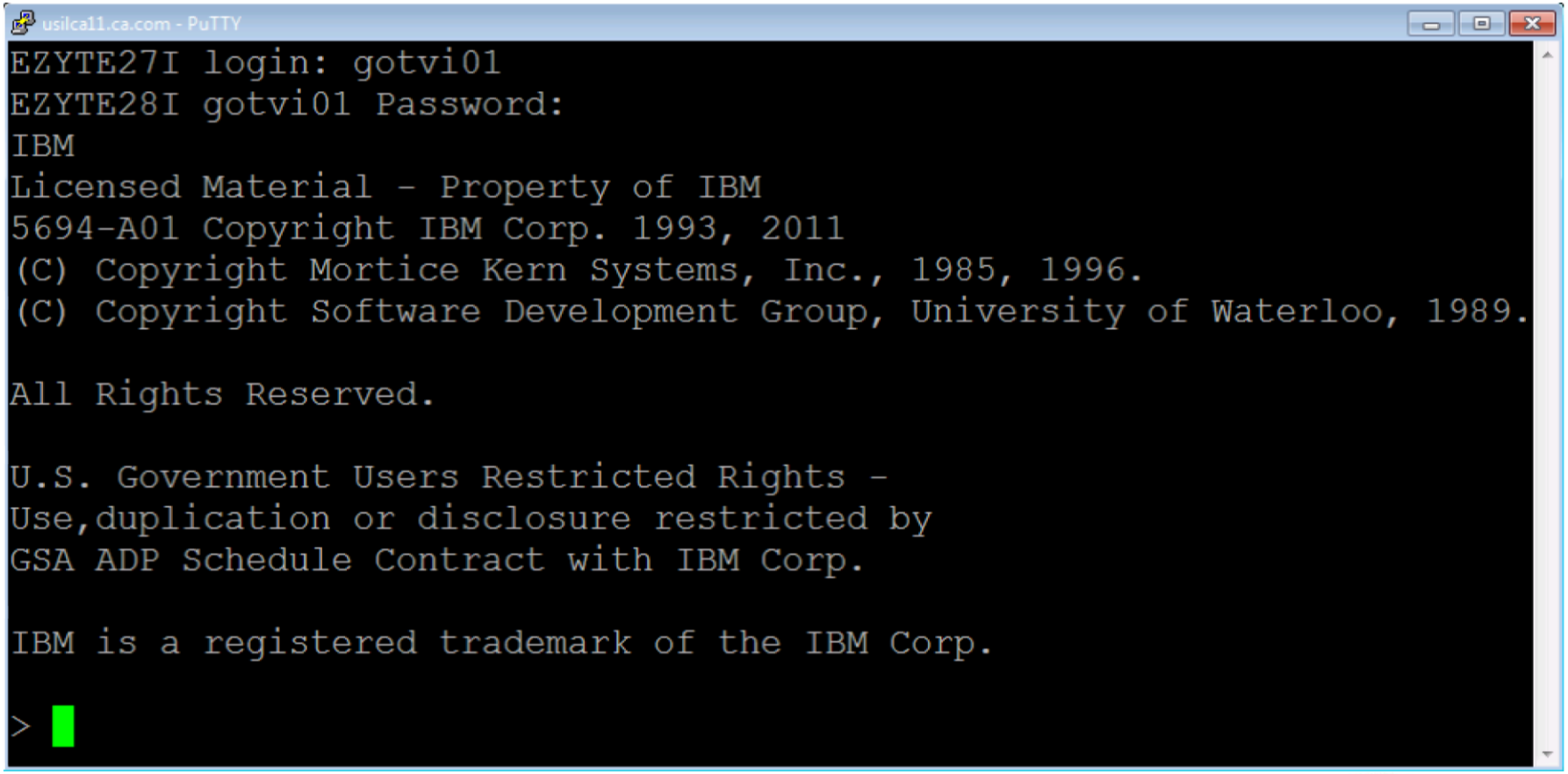


line v.s. character mode UNIX terminal over telnet

- To change the “line discipline”
  - Press “Escape key.”
    - If currently in line mode, press ENTER
  - Watch for “telnet> “ prompt
  - Now type in the command:
    - `mode character`
    - `mode line`
    - to get help `help`
  - Hit ENTER twice

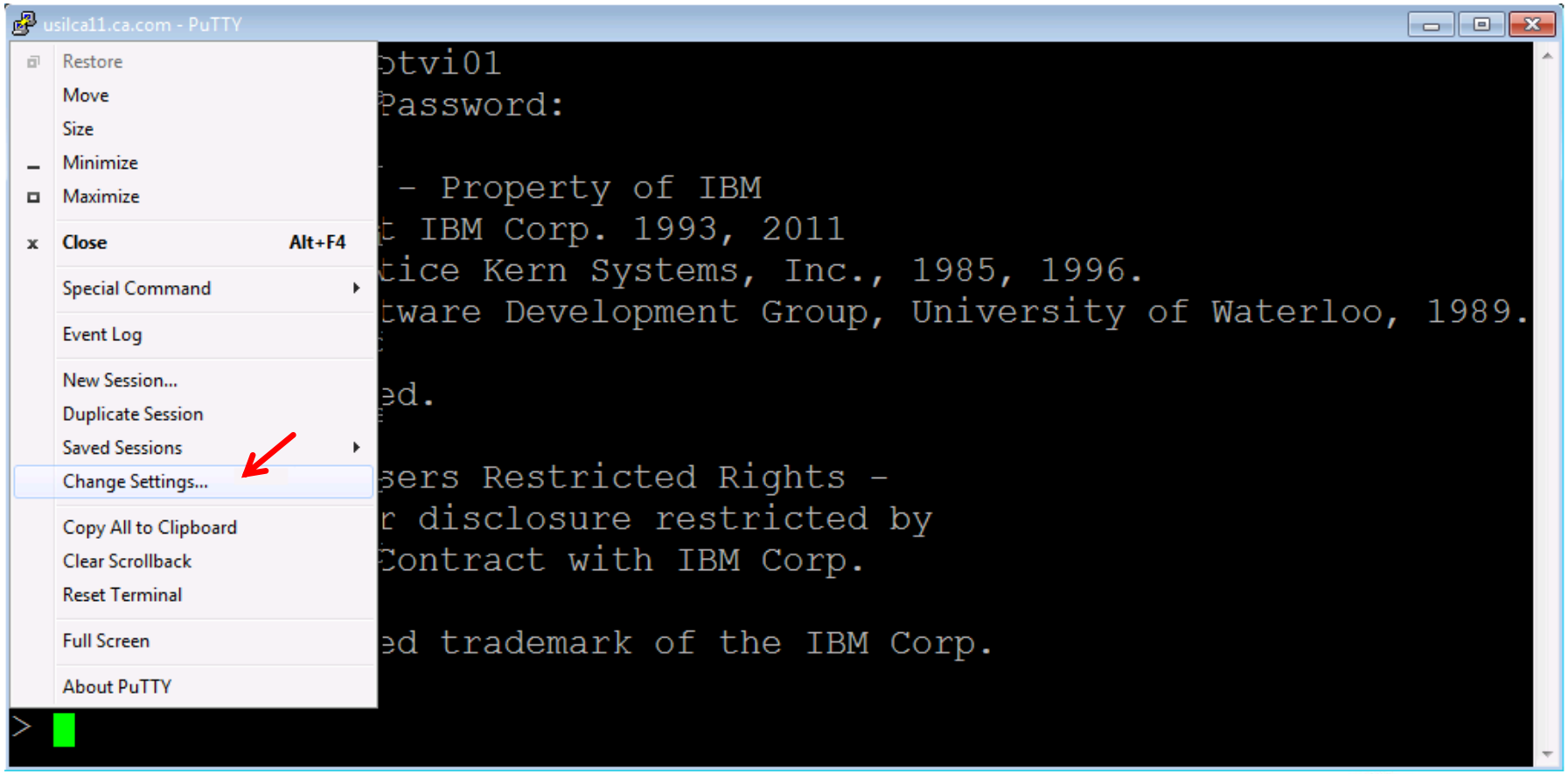
PuTTY
UNIX terminal over telnet

- Standard terminal emulator over telnet for MS Windows is PuTTY

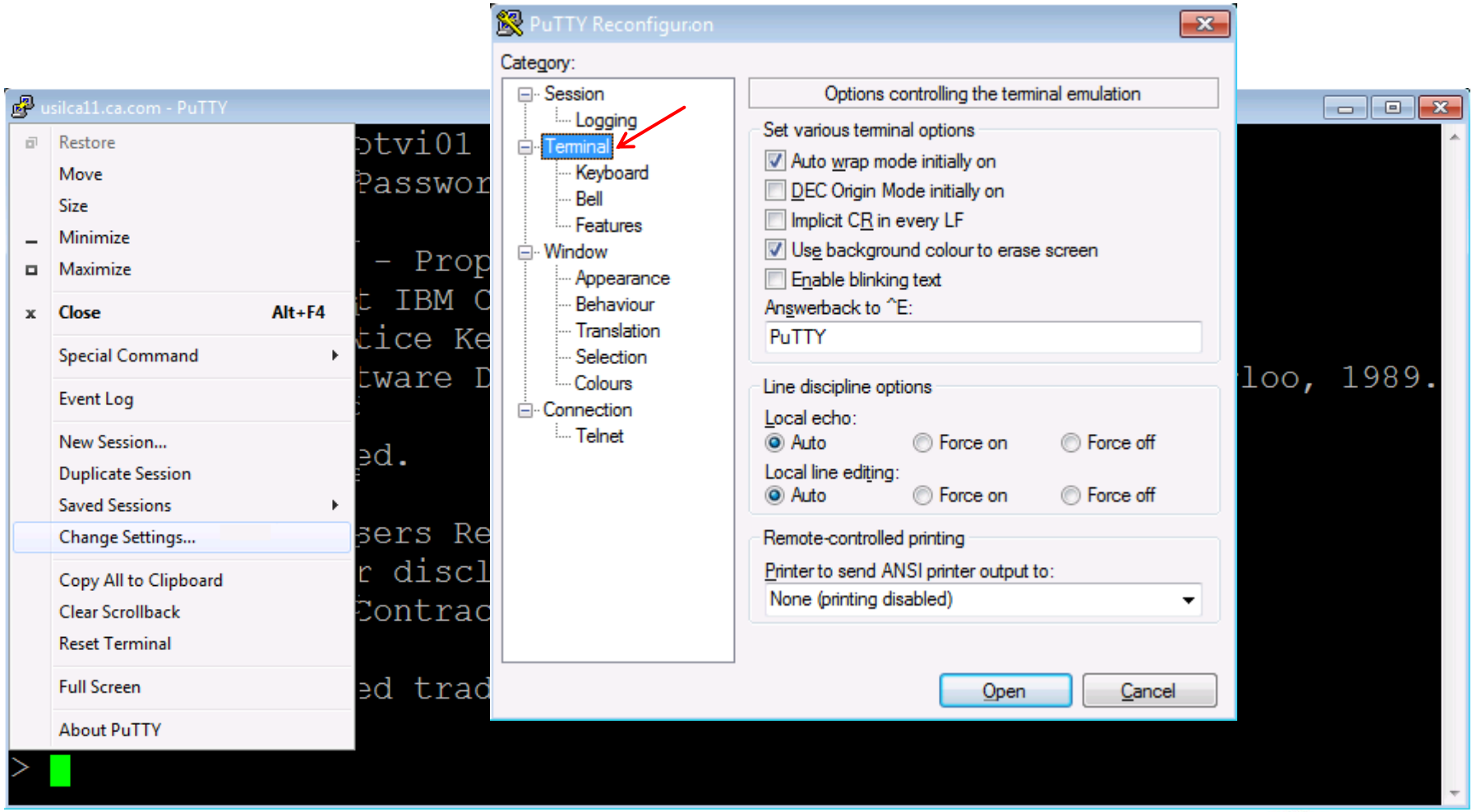


PuTTY - line v.s. character mode
UNIX terminal over telnet

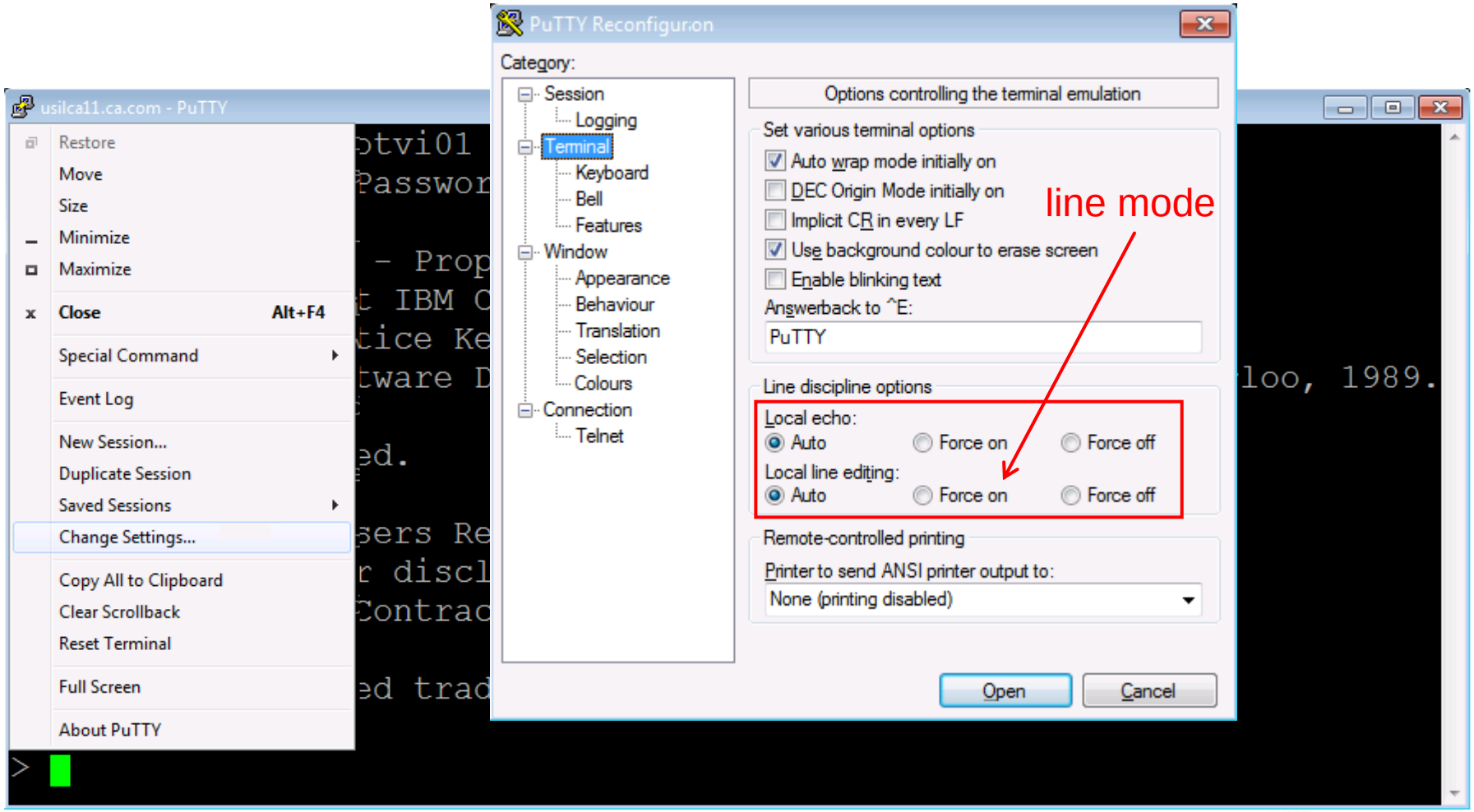
PuTTY - line v.s. character mode UNIX terminal over telnet



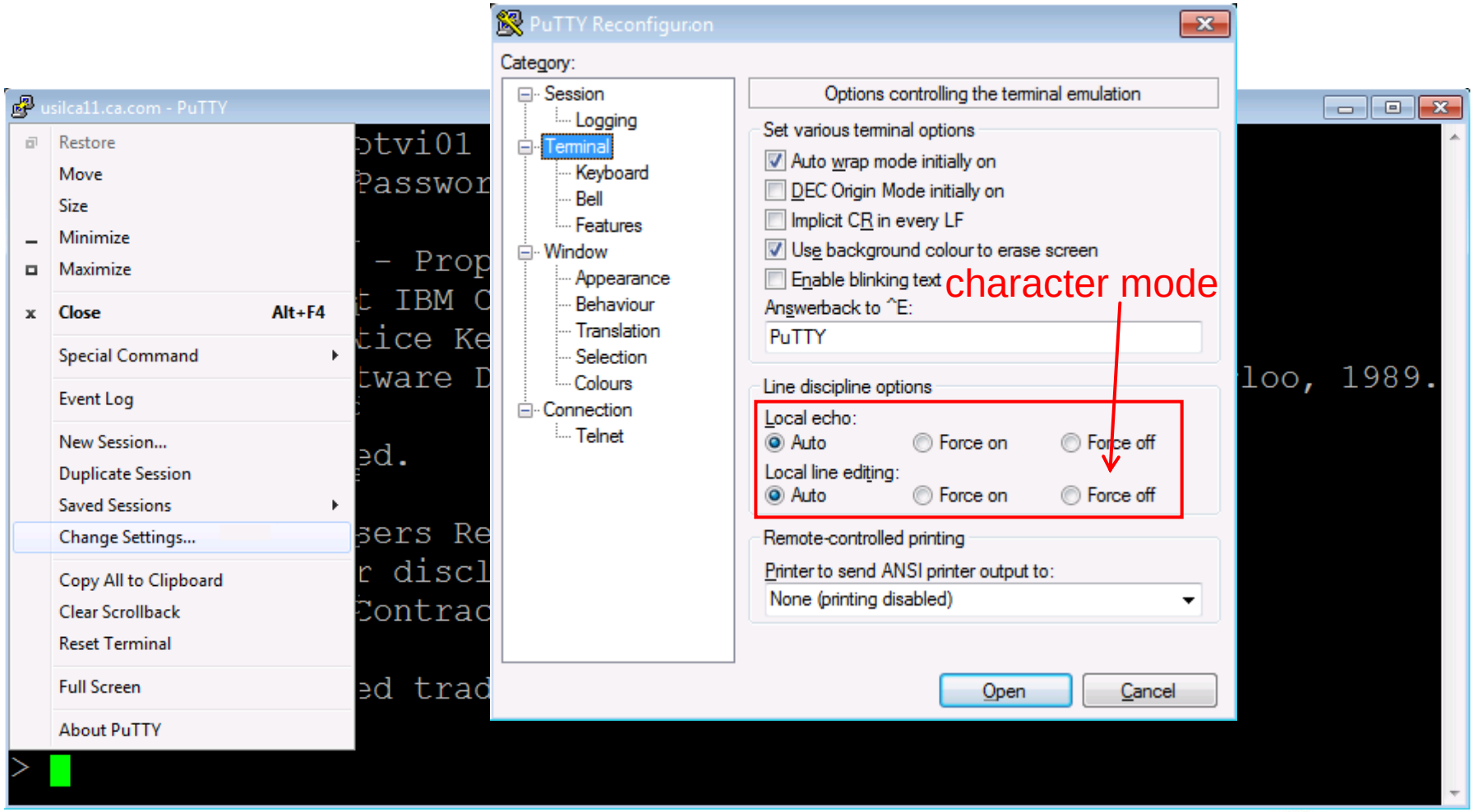
PuTTY - line v.s. character mode
UNIX terminal over telnet



PuTTY - line v.s. character mode
UNIX terminal over telnet



PuTTY - line v.s. character mode
UNIX terminal over telnet



SSH
UNIX terminal over secure shell

- Same appearance and functionality as telnet, but
  - Connection is **encrypted**
  - Operates only in character mode
- Supports
  - Private/public key authentication
  - File transfers (via scp command and sftp, or ftps client)
  - Port forwarding (secure tunnel)
  - Multiple sessions via one connection
    - Not supported on MS Windows
  - Remote command execution
    - Execute z/OS UNIX commands on the mainframe from your PC

SSH
UNIX terminal over secure shell

- SSH server on z/OS
  - Port of OpenSSH - a popular open source implementation of SSH, Part of Ported Tools
  - Can use Integrated Cryptographic Service Facility (ICSF) for hardware acceleration
    - V1R2 with APAR OA37278
  - Automatically converts EBCDIC on the mainframe side to ASCII on the user side (and vice versa)
  - Binary file transfer
    - Use sftp or ftps, not supported for scp
  - OpenSSH User's Guide for z/OS

Summary
UNIX terminal

- To work with a remote UNIX system we need
  - Network connectivity to the system
  - UNIX terminal emulator (vt100, xterm)
    - PuTTY when on a Windows machine
    - xterm when in a graphical user interface on a UNIX
    - xterm, rxvt, gnome-terminal, or konsole when in a GUI on Linux
    - (native terminal interface if signed on to Linux or UNIX console)
  - Protocol to communicate with the remote host
    - telnet or ssh
      - *In Linux/Unix implemented as standalone commands*
      - *In Windows included as part of PuTTY*

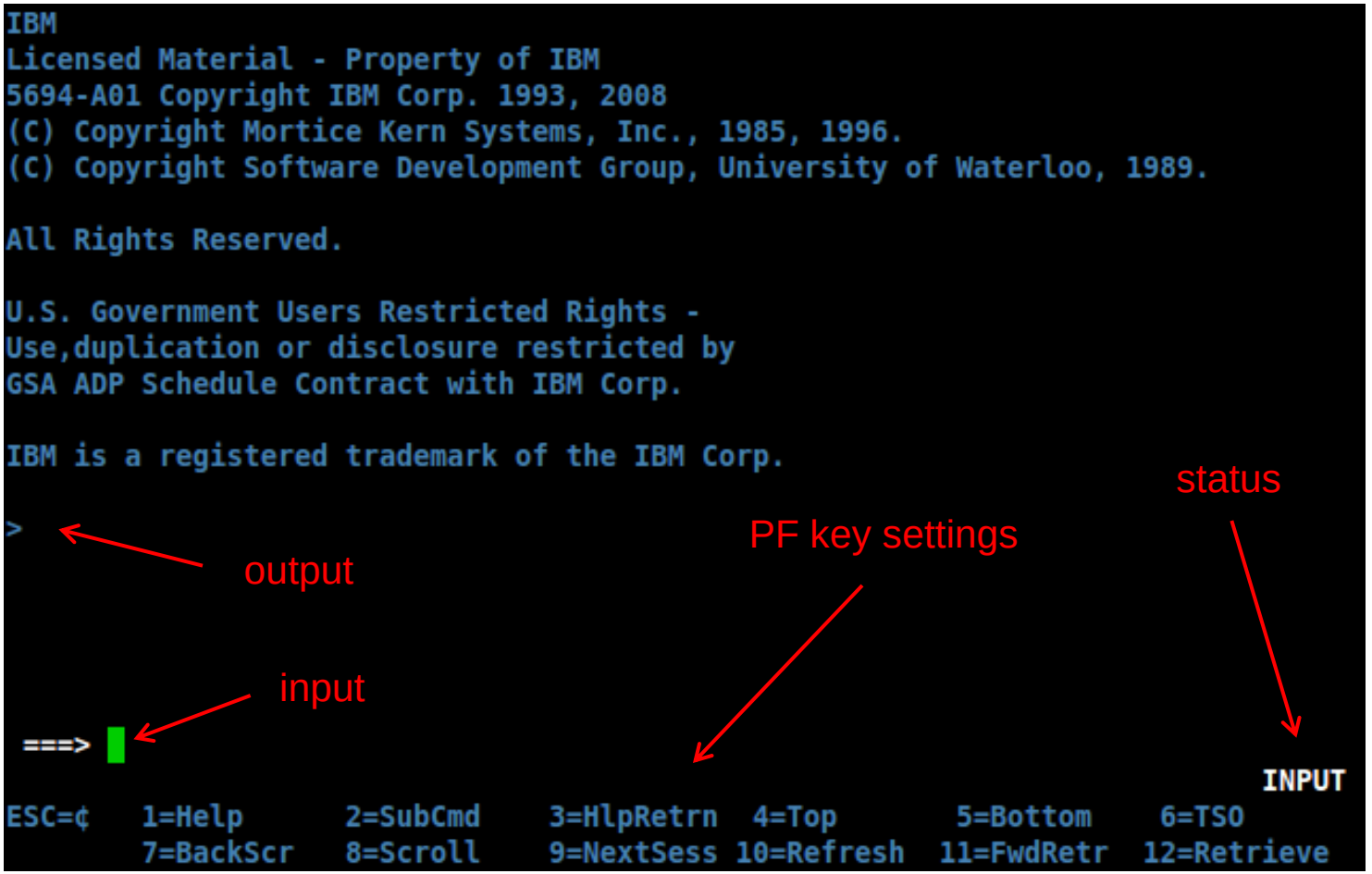
How about running UNIX commands from TSO ?

- Already have connection to the mainframe
- What about the UNIX terminal?
  - Cannot use 3270 directly – not compatible with UNIX
  - Dumb terminal (in the UNIX world)
    - Send and receive data as a sequence of bytes
    - Doesn’t understand cursor and anything related to it
    - Understands no special characters but a newline
- Use 'dumb' terminal and emulate it within TSO
  - This is exactly what OMVS does

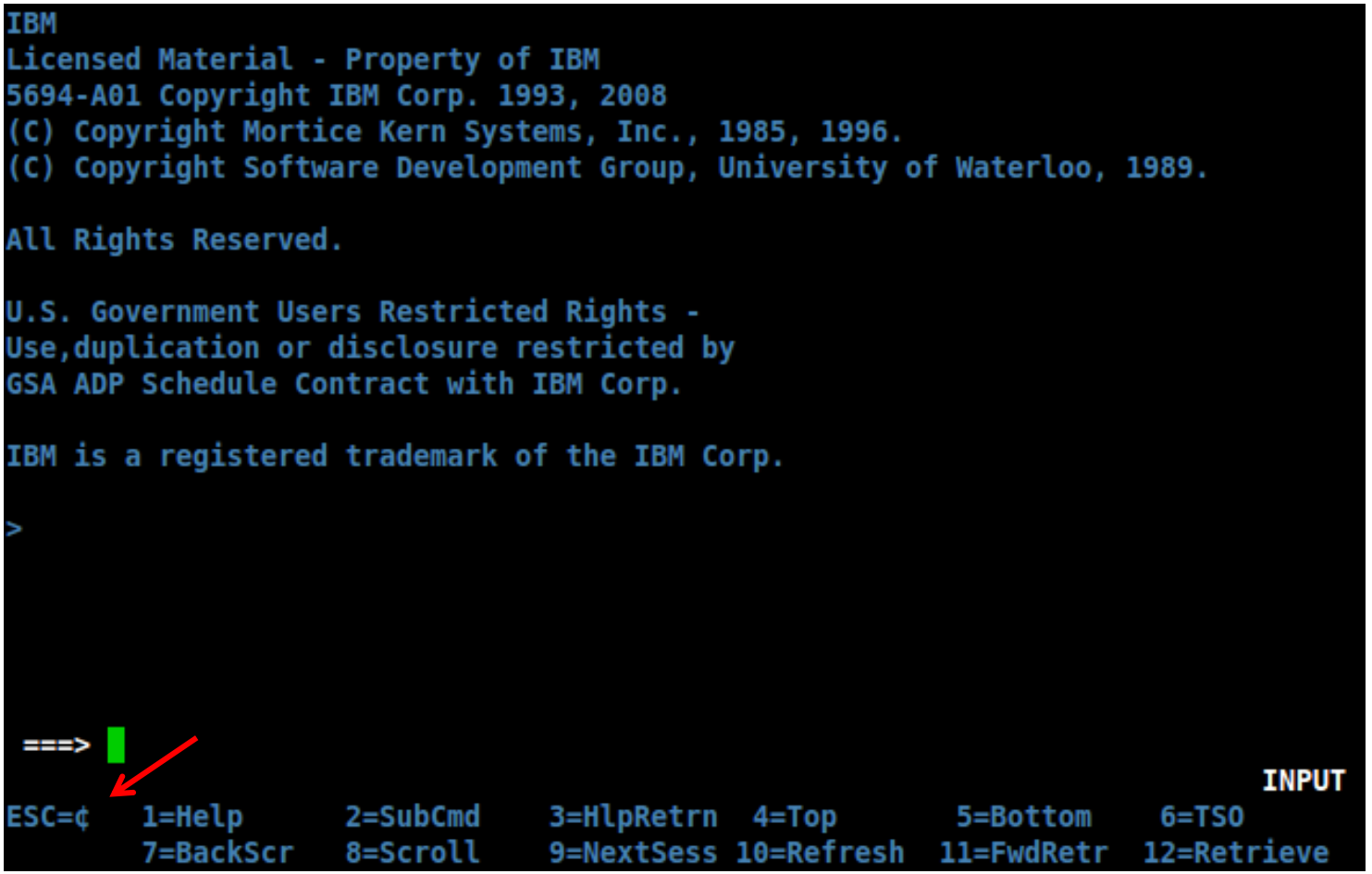
3270 (TSO OMVS)

- Handy if you want to just execute couple commands and look at the output
- Not very practical for longer sessions, I would rather recommend using the UNIX terminal over telnet or SSH
- To start the session enter
  - `OMVS` from TSO command line or `TSO OMVS` from ISPF
- To terminate the session:
  - Type `exit` and hit ENTER
  - hit **`PF2`** (SubCmd), then type in `quit` and hit ENTER
    - Works even during a program execution (e.g. handy if hang)
    - To return back to your session after PF2 type in `return`

3270 (TSO OMVS)



3270 (TSO OMVS)



- Motivation
- Introduction to UNIX
- Interfacing with z/OS UNIX
- **Shells and scripts**
- Copying between files and datasets
- ISHELL

Shells available in z/OS

- z/OS shell – from the Bourne shell family `/bin/sh`
  - implements most scripting features of modern UNIX shells
  - syntax very much compatible with Bash
    - learn from the tutorials and samples available on the web
  - lacks interactive features like command or filename completion
- Bourne Again SHell (Bash) `/sys/s390util/bin/bash`
  - port of GNU Bash - modern shell used in most Linux distributions
  - provided as part of Tools and Toys, not supported
- C shell `/bin/tcsh`
  - syntax similar to C language, nowadays probably obsolete
  - command and filename completion
    - nice for interactive sessions in character mode (line discipline)

Several UNIX commands

- Manipulating files/directories
  - **`cp`** – copy files/directories
  - **`mv`** – move files/directories
  - **`rm`**, **`rm -r (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 input files
  - **`cut`** – print selected columns or fields of every line of the input
- SHARE in Seattle Session 2285, Basic UNIX Shell Commands for the z/OS System Programmer

Simple shell script

- Sample script using GNU diff:

```
#!/bin/sh
# 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"
```

- `/u/users/gotvi01/diff.sh \`
  `file1 file2`

- Sample output

```
--- //'SYS1.PROCLIB(HLASMCL)'
+++ //'SYS1.PROCLIB(HLASMCLG)'
@@ -1,9 +1,10 @@
-//ASMASMCL PROC
+//HLASMCLG PROC
 //*
-//***   ASMASMCL
+//***   ASMASMCLG
 //*
 //* THIS PROCEDURE RUNS THE HI
-//* NEWLY ASSEMBLED PROGRAM
+//* NEWLY ASSEMBLED PROGRAM AN
+//* THE LINK-EDIT IS ACCOMPLIS
 //*
 //C        EXEC PGM=ASMA90,PAR
 //STEPLIB  DD  DSN=ASMA.SASMMO
@@ -24,3 +25,4 @@
 //SYSUT1   DD  DSN=&&SYSUT1,SP
 //            DCB=BUFNO=1
 //SYSPRINT DD  SYSOUT=*
+//G        EXEC PGM=*.L.SYSLMO
```

Shell variables

- **`TMP1="/tmp/$$.1.tmp"`**
  - Set a shell variable
    - **`VAR=value`** (no space allowed around =)
  - Variable name can contain only alphanumeric and _
    - Case-sensitive
  - Value can contain any characters
  - Use variable later on as part of a command
    - **`$VAR, "$VAR"`**
    - to avoid unexpected problems use quotes when spaces in `$VAR`
  - Sample variables available in a script:
    - **`$1 $2 ...`** parameters passed to script from command line
    - **`$$`** process id – often used to generate unique names
    - **`$?`** return code of last command

Environment Variables

- A way of passing information from SHELL to the commands or scripts it executes
  - **`export var`** make shell variable **`var`** an environment variable
  - **`export var=value`** define an environment variable **`var`**
- Examples
  - **`HOME`** – your home directory
  - **`HOST`** – name of the system you are logged on
  - **`PATH`** – where the shell looks for commands; colon separated
- Printing the current values (to the terminal)
  - **`echo $variable_name`**
  - e.g. **`echo $PATH; echo $HOME; echo $HOST`**
  - **`set`** prints all currently defined variables (environment and shell)
  - **`env`** prints all curently defined environtment variables

Input/Output redirection, pipes

- **`cut -c 1-72 "$1" > $TMP1`**
- If no parameters specify files to be processed many commands process the "standard input" (default keyboard)
- Commands print their output to "standard output" and error messages to "standard error" (default terminal screen)
- Shell allows you to override them

`cmd > file`	create / overwrite	**`cmd < file`**	input from a file
`cmd >> file`	create / append	**`cmd << END`** **`line 1`** **`line 2`** **`END`**	input from keyboard terminated by the same word as typed right after <<
`cmd1 | cmd2`	pipe		

Input/Output redirection, pipes

- In special case one program can pass its output as input to another program. This is called a *pipe*
- Redirecting input/output from/to datasets is not directly supported
  - Indirectly via pipes and readmvs/writemvs utilities from Tools and toys, for input also via other commands (**`cat`**, **`cut`**)
- Tools and toys contain many useful utilities
  - **`pdsdir`** – simple rexx script that reads PDS directory and prints it in text to the standard output

```
> pdsdir sys1.proclib
$$$COIBM 04/20/78 07/08/96 13:50      24      19 BELST02
$AGNT    11/13/00 11/13/00 15:09      11      11 CRAAN02
```

Input/Output redirection, pipes

- Imagine we have an MVS utility that
  - Adds members of a PDS into a proprietary dataset format
  - Reads input statements from a SYSIN
  - In order to add a member you have to code these cards:

```
-ADD member,PRMOD
-AUX DD1(member)
-EMOD
```

- Now imagine you have hundreds or thousands of members in your PDS and you want to add them all
  - How would you quickly generate all the input statement for the utility?

Simple shell script using a pipe

- Sample script read from pipe

```
#!/bin/sh
while read member
do
  echo "-ADD $member,PRMOD"
  echo "-AUX DD1($member)"
  echo "-EMOD"
done
```

- Pass output of **pdsdir** to **cut** and then to our script

```
pdsdir sys1.proclib | cut \
-d" " -f 1 | sh mem.sh
```

- Sample output

```
> pdsdir sys1.proclib | cut \
? -d" " -f 1 | sh mem.sh
-ADD $$$COIBM,PRMOD
-AUX DD1($$$COIBM)
-EMOD
-ADD $AGNT,PRMOD
-AUX DD1($AGNT)
-EMOD
-ADD $CHANGES,PRMOD
-AUX DD1($CHANGES)
-EMOD
-ADD $MASAN05,PRMOD
-AUX DD1($MASAN05)
-EMOD
-ADD $SCHD80,PRMOD
-AUX DD1($SCHD80)
-EMOD
```

- Motivation
- Introduction to UNIX
- Interfacing with z/OS UNIX
- Shells and scripts
- **Copying between files and datasets**
- ISHELL

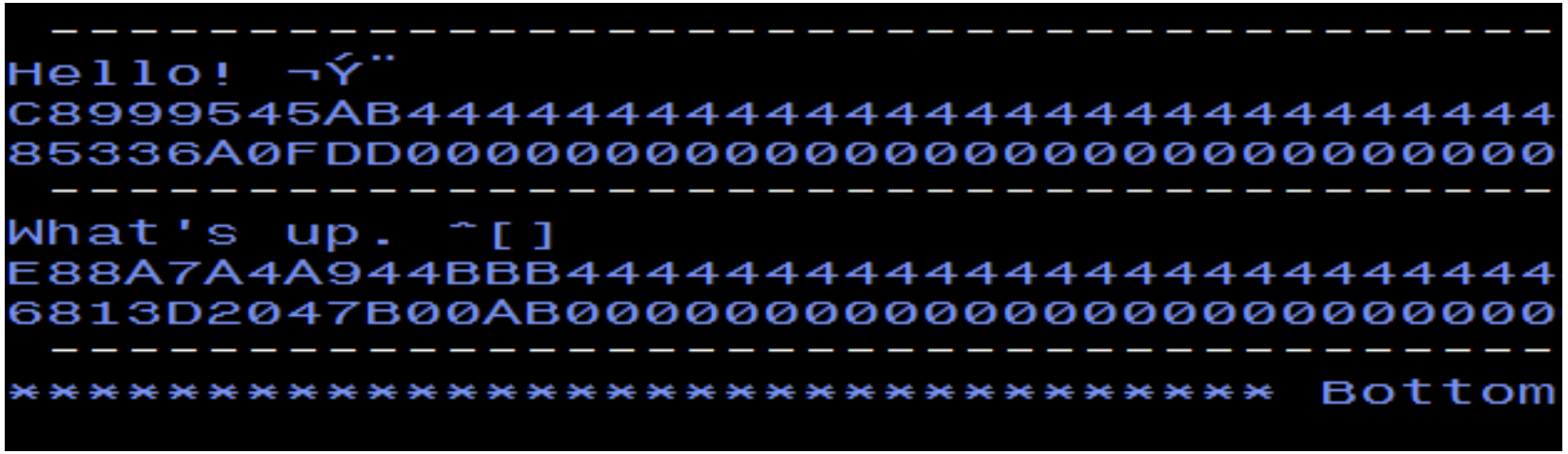
Copying data between files and datasets

- Beware z/OS UNIX files are *byte* oriented, no records / blocks
  - just a stream of bytes
  - in text files end of a line is specified by a "newline" char X'15'
- MVS datasets are record oriented, there is no newline, each record (line) has its length. Depending on the record format the length of each record may vary or be fixed.
  - In fixed format datasets text is usually padded with blanks
- You have to know if you want to replace newlines with trailing spaces or not (and vice versa) when copying back and forth

```
> od -Ax -c -tx1 test
0000000000     H   e   l   l   o   !       ^   [   ]       \n   W   h   a   t
              C8  85  93  93  96  5A  40  5F  AD  BD  40  15  E6  88  81  A3
0000000010     '   s       u   p   .       260 272 273 \n
              7D  A2  40  A4  97  4B  40  B0  BA  BB  15
000000001B
>
```

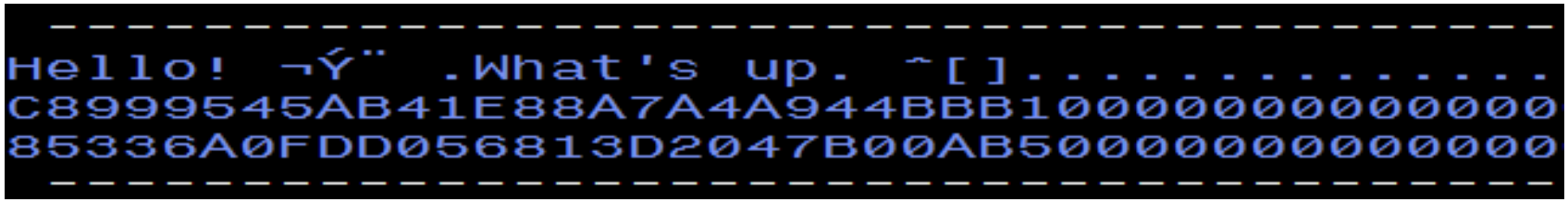
Copying data between files and datasets

- Beware z/OS UNIX files use IBM-1047 as default encoding
  - MVS datasets use IBM-037 as default, these differ in ^ [ ]
  - When copying between files and datasets you have to know if you want to convert the encoding or not (e.g. C source code)



Copying data between files and datasets

- From the shell use **cp**
- Copy a text file to a sequential dataset
  - **cp -T source_file "//'hlq.desti(nation)'"**
- Copy a binary file to a sequential dataset
  - **cp -B source_file "//'hlq.desti(nation)'"**
- Copy an executable binary (a program object) to a PDS/E
  - **cp -X source_pgm "//'hlq.desti(nation)'"**
- For more details see usage notes and examples



Copying data between files and datasets

- 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)

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
//FILE       DD PATH='/tmp/a'
//DTST       DD DISP=SHR,DSN=GOTVI01.JCL(A)
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
OCOPY INDD(FILE) OUTDD(DTST) TEXT
/*
```

Converting data between character sets

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

- Sample usage

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

this will covert file *infile* in code page 037 to code page 1047 and save the output in file *outifle*

FTP
Converting data between character sets

- remote access to
  - UNIX file system

    ```
    ascii
    get /u/users/gotvi01/a.txt
    ```

  - datasets

    ```
    ascii
    quote site ispfstats
    quote site sbdataconn=
            (ibm-037, iso8859-1)
    put ab.txt 'gotvi01.txt(ab)'
    ```

  - JES

    ```
    quote site filetype=jes
    get JOB1234.1
    ```

- **quote help site**

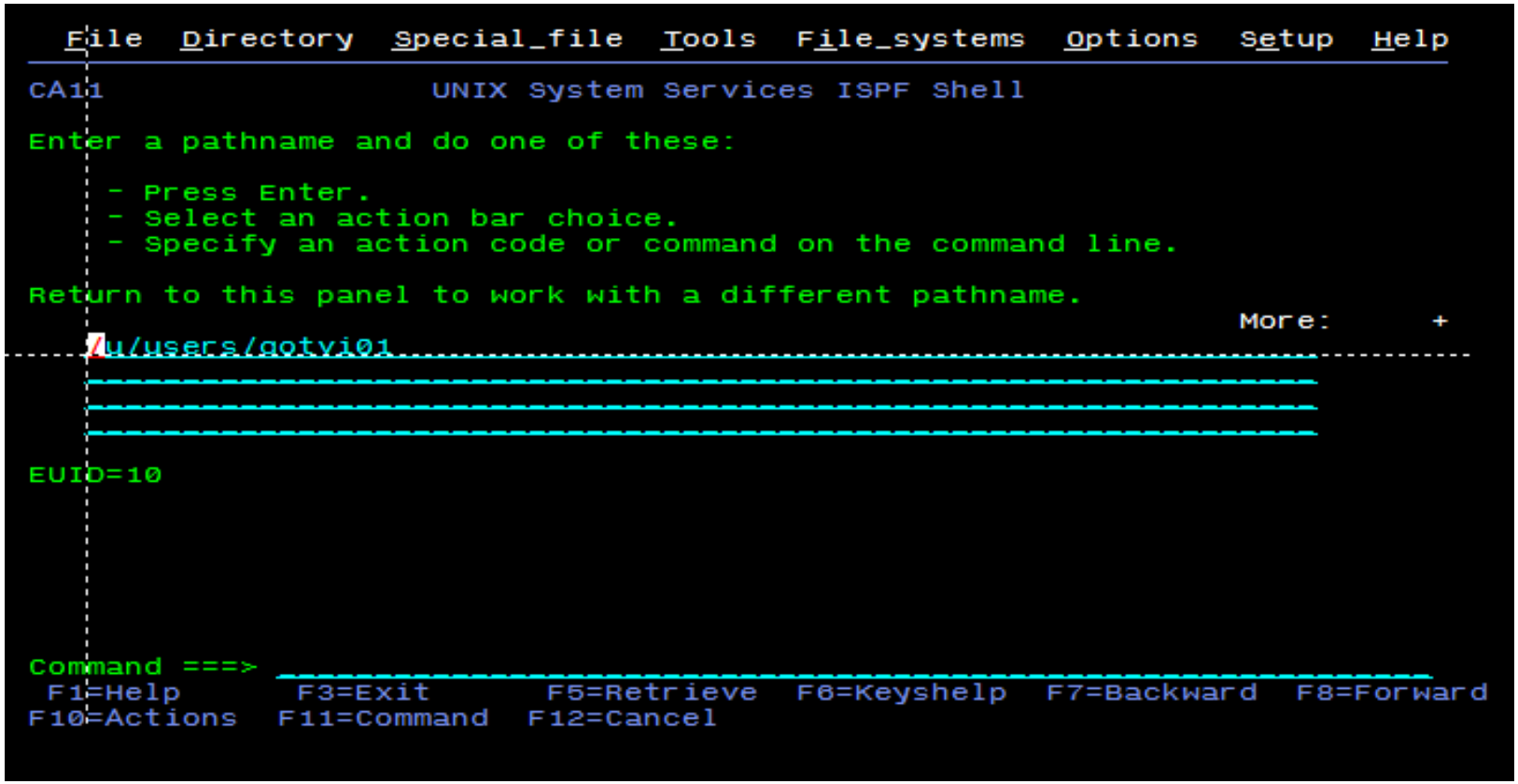
```
james:~ vitgottwald$ ftp xg56
Connected to xg56.ca.com.
220-FTPD011 IBM FTP CS V1R10 at XG56.CA.
220 Connection will close if idle for mo
Name (xg56:vitgottwald): gotvi01
331 Send password please.
Password:
230 GOTVI01 is logged on.  Working direc
Remote system type is MVS.
ftp> quote site filetype=jes
200 SITE command was accepted
ftp> quote site jesjobname=y8v*
200 SITE command was accepted
ftp> dir
229 Entering Extended Passive Mode (|||3
125 List started OK for JESJOBNAME=Y8V*,
JOBNAME  JOBID    OWNER    STATUS CLASS
Y8VDIFF  JOB00298 GOTVI01  OUTPUT A
Y8VGREP  JOB00296 GOTVI01  OUTPUT A
Y8VBASH  JOB00262 GOTVI01  OUTPUT A
250 List completed successfully.
ftp> get JOB00262.1
local: JOB00262.1 remote: JOB00262.1
229 Entering Extended Passive Mode (|||3
125 Sending data set GOTVI01.Y8VBASH.JOB
1689      55.54 KB/s
250 Transfer completed successfully.
1689 bytes received in 00:00 (14.01 KB/s
ftp> quit
221 Quit command received. Goodbye.
```

- Motivation
- Introduction to UNIX
- Interfacing with z/OS UNIX
- Shells and scripts
- Copying between files and datasets
- **ISHELL**
- Running z/OS UNIX tools in batch

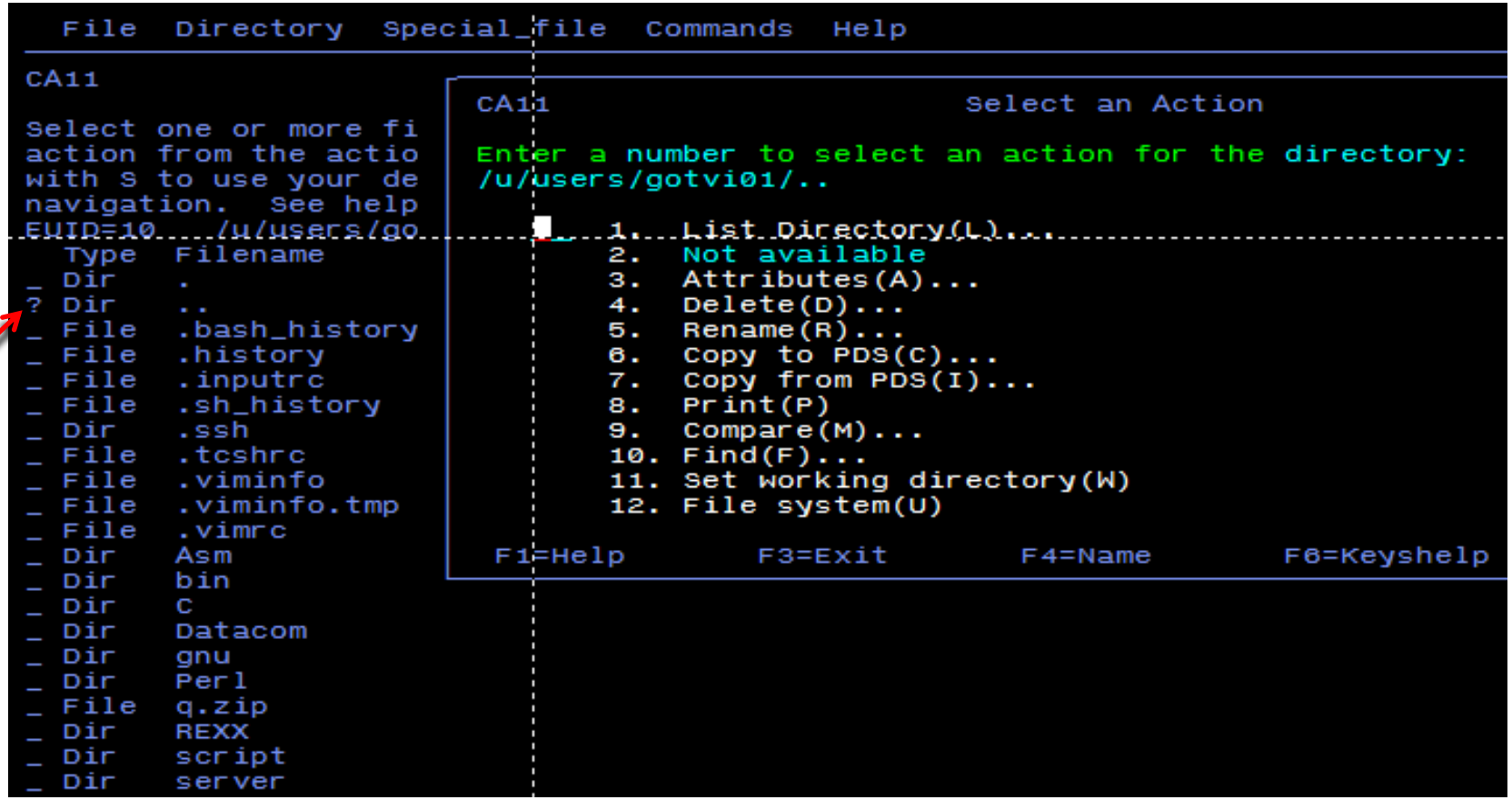
ISHELL

- ISPF Shell (ISHELL)
  - Not really a UNIX shell but rather a
  - Set of ISPF panels to let you do what is traditionally done in a shell but in the comfort of a panel interface
  - Useful mainly for
    - File system navigation and administration
    - File browsing, editing
  - Typically a "must have" for MVS veterans or anyone avoiding the "geeky" command line ☺
  - Today many ISPF panels let you type in directly a path to a file in place of a dataset name
  - `ISHELL` from TSO or `TSO ISHELL` from ISPF

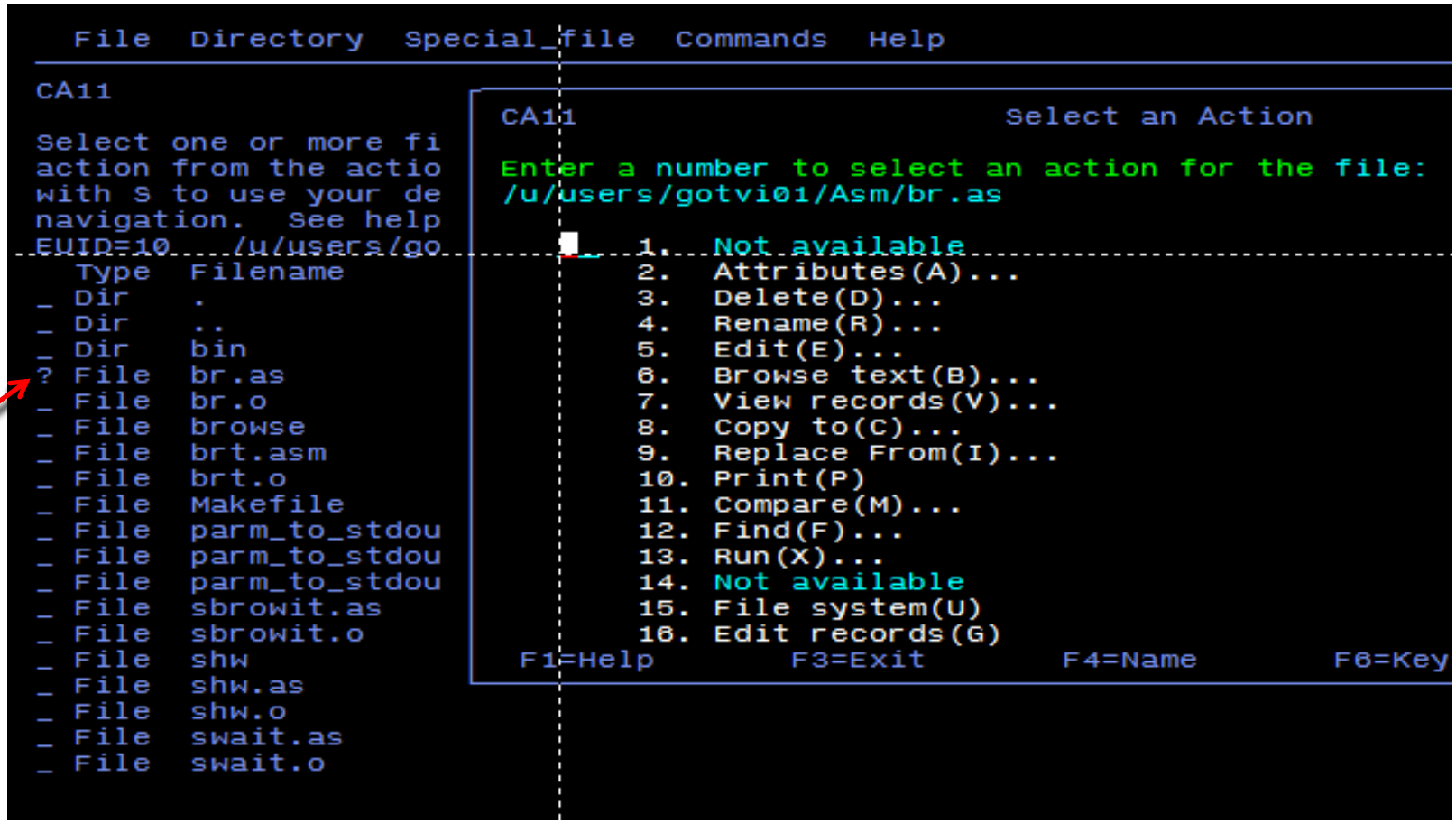
ISHELL


File Directory Special_file Tools File_systems Options Setup Help
CA11 UNIX System Services ISPF Shell
Enter a pathname and do one of these:
- Press Enter.
- Select an action bar choice.
- Specify an action code or command on the command line.
Return to this panel to work with a different pathname.
More: +
/u/users/gotvi01
EUID=10
Command ===>
F1=Help F3=Exit F5=Retrieve F6=Keyshelp F7=Backward F8=Forward
F10=Actions F11=Command F12=Cancel

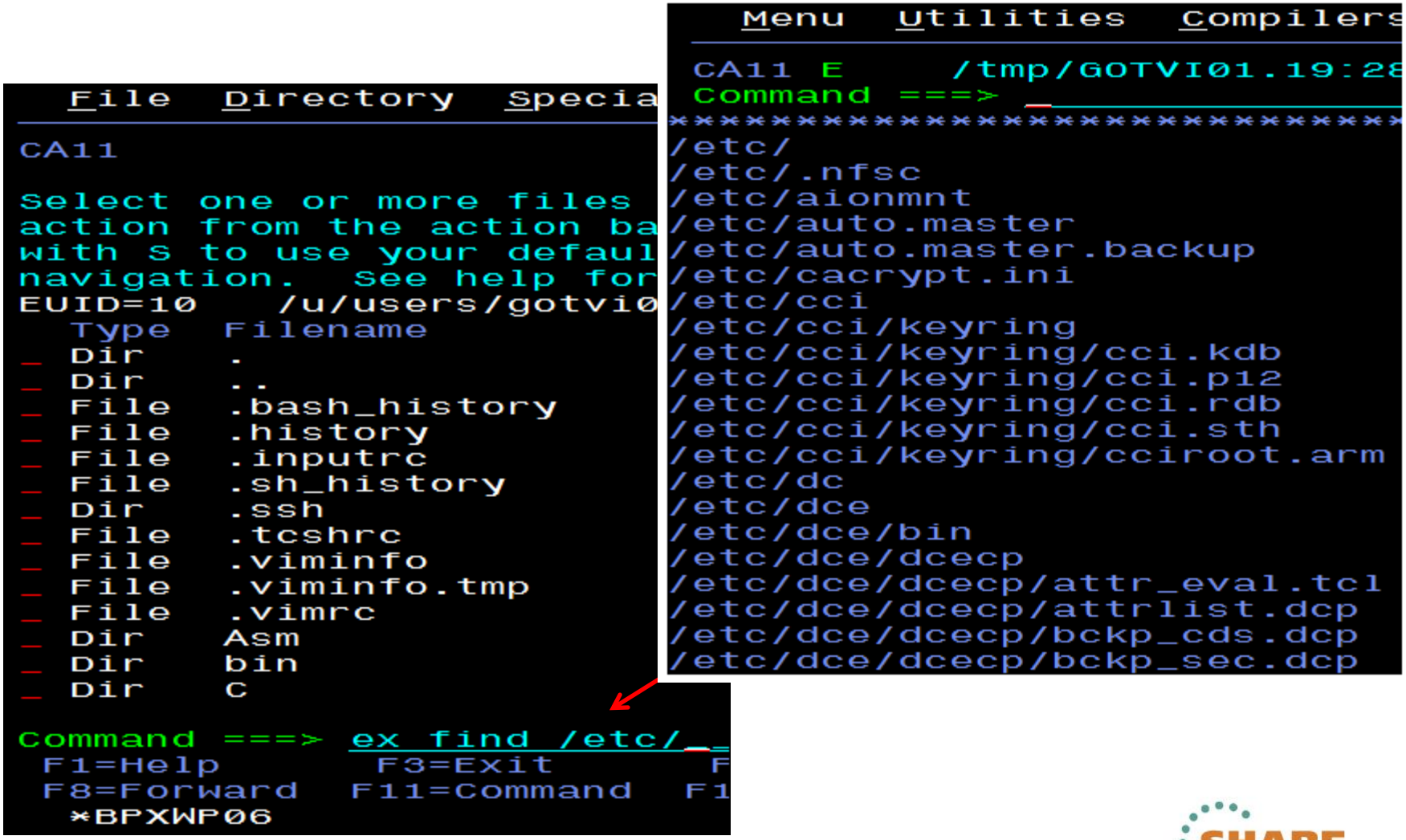
ISHELL - directory options



ISHELL - file options



ISHELL - executing commands from



- Motivation
- Introduction to UNIX
- Interfacing with z/OS UNIX
- Shells and scripts
- Copying between files and datasets
- ISHELL

Thank you for your attention!

Please do not forget to fill in evaluation forms.

Resources: SHARE sessions

- Previous SHARE conferences
  - Basic UNIX Shell Commands for the z/OS System Programmer, Share in Seattle
  - TSO OMVS and You: What to Make of UNIX System Services, Share in Orlando
  - z/OS UNIX File System Administration, Share in Anaheim
  - z/OS Basics: z/OS UNIX Shared File, Share in Orlando
  - Killing Zombies, Breaking Pipes, and other UNIX Shenanigans, SHARE in Atlanta
  - UNIX Systems Services in Today's Mainframe Software, SHARE in Atlanta

Resources: IBM z/OS UNIX manuals

- z/OS V1R13.0 UNIX System Services
  - GA22-7800-19 Planning
  - SA22-7801-14 User's Guide
  - SA22-7802-14 Command Reference
  - SA22-7807-12 Messages and Codes
  - … and more … http://publib.boulder.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.r13.bpx/bpx.htm

Resources: Web references

- Community support – MVS-OE mailing list
  http://vm.marist.edu/archives/mvs-oe.html

- UNIX Tutorial for Beginners
  http://www.ee.surrey.ac.uk/Teaching/Unix/
- UNIX TOOLBOX – a pool of typical usage examples
  http://cb.vu/unixtoolbox.xhtml

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

Resources: Legacy UNIX books

- The UNIX Programming Environment, Brian W. Kerninghan, Rob Pike, 1984

- UNIX Programmer's Manual, 7th Edition, Bell Telephone Laboratories, Incorporated, 1979

- Motivation
- Introduction to UNIX
- Interfacing with z/OS UNIX
- Shells and scripts
- Copying between files and datasets
- ISHELL
- **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
  - BPXBATCH
    - 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
  - BPXBATSL
    - Does a “local spawn”, runs the commands in its address space
- For documentation see
  - *z/OS UNIX System Services Command Reference*
- Also see Dovetail’s Co:Z Batch for more options

Running in UNIX tools 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)'"
/*
```

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

Accessing datasets

- 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
  - **`pdsdir`** – utility for listing members of a PDS
    - This is a REXX script, not a C program, the argument is just DSN without any // or apostrophes (') or quotes (") around it

Running in batch II

- 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
- Remember that C compiler produces LE compliant prgrms!

```
//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.SCCNPRC(CBCC)
//IN2        DD DISP=SHR,DSN=CBC.SCCNPRC(CBCCL)
```

Language Environment options, separator, parameters passed to the program

Running in batch, 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
JRFileNotThere: The requested file does not exist

Action: The service cannot be performed unless the named
file exists.
```

Running in batch, tips

- 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).
- Make sure there are no record numbers in positions 73-80 of input files processed by UNIX utilities. Especially when your commands come from an inline dataset! UNIX tools do not ignore these positions. They interpret them like any another parameter and then usually fail (e.g. file not found).
- 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.

Caling programs residing in a dataset from a shell

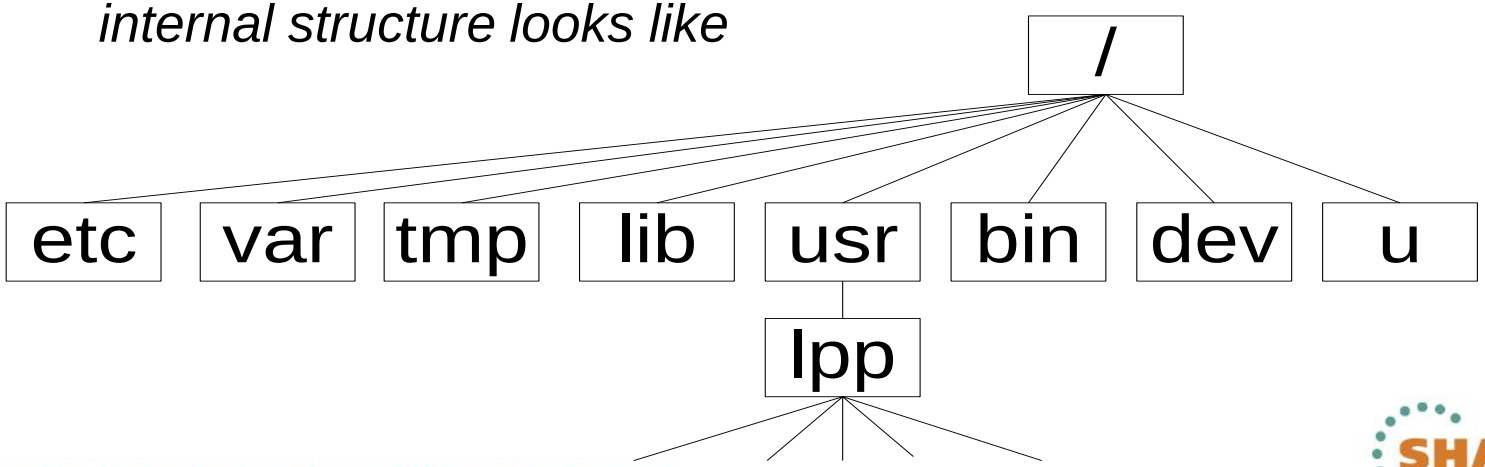
- STEPLIB Environment Variable
  - Colon-separated list of dataset names, e.g. **`STEPLIB=DSN1:DSN2:DSN3`**
  - When you run a command and STEPLIB variable is not empty, STEPLIB DD within the address space is allocated based on the contents of the variable
  - This allows you to link to MVS programs residing in the datasets

- External symbolic link
  - Created via **`ln -e "//'dataset_name(module)'" link_name`**

```
> ls -l
total 672
erwxrwxrwx   1 GOTVI01  OMVSGRP       15 Aug  8 04:01 status2 -> //load(status2)
>
```

z/OS UNIX file system

- UNIX file system is a hierarchical directory sturcture
  - Starts with a root "/"
  - Every file or a directory contained in its "parent" directory ".."
    - Parent directory of root is root
    - Files are just streams of bytes
      - *No internal structure from the operating system perspective*
      - *Application working with a file has to understand what the internal structure looks like*



Navigating the directory structure

- Described in every tutorial
- Concept of a “current working directory” or “.”
- Concept of a “home directory” or “~”
- “parent directory” or “..”

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

z/OS UNIX System Services
The Compiler
dbx
Forums
Library
Education
Links
Release History
IBM Ported Tools for z/OS
Porting Applications
Tools & Toys
APARs
Performance
Presentations
Contact z/OS UNIX

Related links
- Global Solutions Directory

Tools and toys

Most packages listed here are free (some are pointers to third party packages which may not be free). There are no warranties of any kind, and there is no service or technical support available for these from IBM.

Links to Tools and Toys

Due to the large volume of tools and toys, we've had to break the tools and toys page into separate sections.

Ported Tools

Apache, ACE, atools, autoconf, automake, bash, bison, bzip2, cpost, diffutils, emacs, findutils, flex, gmake, gnats, GNU grep, groff, gsar, gzip, host, hostname, id-utils, ksh93, less, libtool, lynx, m4, nedit, openssh, openssl, patch, perl, PGP, rcs, sha, the, vim, unzip, webalizer, zip

UNIX Tools

auditid, booksrvcar, Bruno's Bag of Tools, bpxget, bpxput, bpxstop, bpxwdyn, bpxwtabs, cksparse, colonies, COBOL compile environment, db2pb, db2imsrx, describe, dirsize, execsql, ext, fduse, finuse, fsadmin, fsinuse, fsview, getuids, ICS2WLM, IOSSPOFD, i, ifind, ihsCMT, IPLSTATS, isu, Java OMTA, KEYXFER, libascii, makedep, mcp, MSGLG610, MVSJob, odb2, oeconsol, oesvp, omvstape, ORACOMP, PAV Analysis Tool, passwd, pdsdir, perr, pschart, rac, rangelks, readmvs and writemvs, rexxfunc, rexx compiler utility, RTOOLS, sdsf, seff, setstack, stat, startd, submit, submit2, SVC analyzer, sysout, tcpcfg, tsocmd, USStools, uurestore, viascii, view, websearch, wjsigshl, zFS Large Directory Utility, zlsof

What's new

- IPLSTATS - contains two programs which can be used to report on the time taken by z/OS to initialize various parts of the operating system (January 8, 2010)
- submit2 - a shell command that submits JCL, waits until the job is done and retrieves the job output. (December 15 2009)
- zFS Large Directory Utility - this utility identifies zFS directories that might potentially cause a performance issue.(September 3, 2009)
- MSGLG610 - the

Filesystem security (access rights)

- Command `ls -l` displays many file attributes

```
> ls -l
total 158
drwxr-xr-x    6 STCSYS   OMVSGRP      544 Sep  6  2010 SYSTEM
dr-xr-xr-x  214 STCSYS   TTY            0 Aug  2 17:17 a
drwxr-xr-x    2 STCSYS   OMVSGRP      256 Dec  1  2010 agent
drwxr-xr-x    4 STCSYS   OMVSGRP    16384 Apr 25 13:48 bin
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 11  2010 dev -> $SYSNAME/dev
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 12  2010 etc -> $SYSNAME/etc
lrwxrwxrwx    1 STCSYS   OMVSGRP       16 Oct 11  2010 krb5 -> etc/dce/var/krb5
drwxr-xr-x    2 STCSYS   OMVSGRP      288 Sep 25  2010 lib
drwxr-xr-x    2 STCSYS   OMVSGRP      352 Sep  6  2010 opt
drwxrwxrwx    6 PIFAI01  FRAMEWKG    8192 May 27 05:19 s
drwxr-xr-x    4 STCSYS   OMVSGRP     2432 Sep 25  2010 samples
drwxr-xr-x   74 STCSYS   OMVSGRP     8192 Jul 13 22:11 sys
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 11  2010 tmp -> $SYSNAME/tmp
drwxr-xr-x    5 STCSYS   OMVSGRP      352 Dec  1  2010 u
drwxr-xr-x   12 STCSYS   OMVSGRP      576 Sep  6  2010 usr
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 12  2010 var -> $SYSNAME/var
>
```

Filesystem security (access rights)

- Command `ls -l` displays among others *symbolic links*
- There is session 9875: z/OS UNIX Shared FileSystem ...

```
> ls -l /
total 158
drwxr-xr-x    6 STCSYS   OMVSGRP      544 Sep  6  2010 SYSTEM
dr-xr-xr-x  214 STCSYS   TTY            0 Aug  2 17:17 a
drwxr-xr-x    2 STCSYS   OMVSGRP      256 Dec  1  2010 agent
drwxr-xr-x    4 STCSYS   OMVSGRP    16384 Apr 25 13:48 bin
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 11  2010 dev -> $SYSNAME/dev
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 12  2010 etc -> $SYSNAME/etc
lrwxrwxrwx    1 STCSYS   OMVSGRP       16 Oct 11  2010 krb5 -> etc/dce/var/krb5
drwxr-xr-x    2 STCSYS   OMVSGRP      288 Sep 25  2010 lib
drwxr-xr-x    2 STCSYS   OMVSGRP      352 Sep  6  2010 opt
drwxrwxrwx    6 PIFAI01  FRAMEWKG    8192 May 27 05:19 s
drwxr-xr-x    4 STCSYS   OMVSGRP     2432 Sep 25  2010 samples
drwxr-xr-x   74 STCSYS   OMVSGRP     8192 Jul 13 22:11 sys
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 11  2010 tmp -> $SYSNAME/tmp
drwxr-xr-x    5 STCSYS   OMVSGRP      352 Dec  1  2010 u
drwxr-xr-x   12 STCSYS   OMVSGRP      576 Sep  6  2010 usr
lrwxrwxrwx    1 STCSYS   OMVSGRP       12 Oct 12  2010 var -> $SYSNAME/var
>
```

Filesystem security (access rights), OBROWSE

```
 Menu  Utilities  View  Options  Help
CA11                    z/OS UNIX Directory List
Command ===>

Pathname . : /u/users/gotvi01

Command  Filename        Message              Type Permission
---------------------------------------------------------------
_______ .                                     Dir  rwxr-xr-x
_______ ..                                    Dir  rwxr-xr-x
_______ .bash_history                         File rw-------
_______ .history                              File rw-------
_______ .inputrc                              File rw-rw-rw-
_______ .sh_history                           File rw-------
_______ .ssh                                  Dir  rwxr-xr-x
_______ .tcshrc                               File rwxr--r--
_______ .viminfo                              File rw-------
_______ .viminfo.tmp                          File rw-------
_______ .vimrc                                File rw-rw----
_______ bin                                   Dir  rwxrwxrwx
_______ gnu                                   Dir  rwxr-xr-x
_______ q.zip                                 File rw-rw----
_______ script                                Dir  rwxrwxrwx
_______ server                                Dir  rwxrwxr--
_______ slavek                                Dir  rwxrwxrwx
_______ svc.txt                               File rw-rw-rw-
_______ s3270                                 Dir  rwxrwxrwx
_______ test                                  File rwx------
_______ toys                                  Dir  rwxrwxrwx
_______ Asm                                   Dir  rwxrwxrwx
_______ C                                     Dir  rwxr-xr-x
_______ Datacom                               Dir  rwxrwxrwx
_______ Perl                                  Dir  rwxr-xr-x
_______ REXX                                  Dir  rwxrwxrwx
_______ Unix                                  Dir  rwxrwxrwx
***************************************************************
```

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 provied 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://publib.boulder.ibm.com/infocenter/zos/v1r12/topic/com.ibm.zos.r12.bpxb200/comp.htm, a short excerpt follows …

Comparing UNIX, MVS, and z/OS UNIX security

Category	Traditional UNIX	MVS	z/OS UNIX
User identity	Users are assigned a unique UID, a 4-byte integer and user name.	Users are assigned a unique user ID of 1-to-8 characters.	Users are assigned a unique user ID with an associated UID.
Security identity	UID	User ID	UID for accessing traditional UNIX resources and the user ID for accessing traditional z/OS resources

Comparing UNIX, MVS, and z/OS UNIX security

Category	Traditional UNIX	MVS	z/OS UNIX
Login ID	Name used to locate a UID	Same as the user ID	Same as the user ID
Special user	Multiple user IDs can be assigned a UID of 0.	RACF® administrator assigns necessary authority to users.	Multiple user IDs can be assigned a UID of 0 or users can be permitted to BPX.SUPERUSER.

Comparing UNIX, MVS, and z/OS UNIX security

Category	Traditional UNIX	MVS	z/OS UNIX
Identity change from regular user to superuser	The **su** shell command allows change if user provides password for the root. Password phrases are not used in traditional UNIX security.	No provision for unauthorized user to change identity.	The **su** 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.
Terminate user processes	Superuser can kill any process.	MVS operator can cancel any address space.	Superuser can kill any process.

Copying data over network

- Over network
  - FTP
  - Open SSH (SFTP, FTPS,scp), part of Ported Tools
  - cURL – great utility originally for downloading and uploading data over HTTP, FTP, FILE and even more protocols, part of Supplementary Toolkit for z/OS

Compiling UNIX software packages

- 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

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)
  - `'\\ *$'` – match lines ending with a back-slash (that may or may not be followed by blanks)

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 z/OS UNIX "Ported Tools"
- Supports catalog search through the –r option (//'hlq.mlq*')
- Following command searches SYS1.MACLIB for all lines containing an SVC 34 instruction according to the logic HLASM uses to process its source code

```
/sys/s390util/bin/grep -ir \
'^\([.]\?[@#$a-z0-9]\+\)\? \+svc \+34' \
"//'sys1.maclib'"
```

GNU grep cont'd

```
> /sys/s390util/bin/grep -ir '^\([.]\?[@#$a-z0-9]\+\)\? \+svc \+34 ' "//'sys1.maclib'"
//'sys1.maclib(MGCR    )':.ISVC    SVC   34                              ISSUE MGCR SVC
//'sys1.maclib(MGCRE   )':         SVC 34                             ISSUE SVC 34
//'sys1.maclib(QEDIT   )':         SVC   34 -                  INVOKE COMMAND SCHEDULER
>
```

UNIX tools and their MVS counter parts

- 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

UNIX	MVS
cp	IEBGENER,IEBCOPY
mv, rm	IDCAMS
diff	SuperC/SuperCE
grep	Search-For/Search-ForE
pax	TRSMAIN
shell scripts, commands	REXX execs + ISPF services
find, locate	Catalog Search Interface + LISTDS