
z/OS UNIX System Services Security 50.5

(Only half as introductory as UNIX Security 101!)

RACF Users Group of New England
June 4, 2015

Bruce R. Wells
brwells@us.ibm.com



Trademarks

- See url <http://www.ibm.com/legal/copytrade.shtml> for a list of trademarks.



Agenda

- What makes UNIX UNIX?
- What makes a bunch of MVS data sets a UNIX file system?
- What makes a RACF user a UNIX user? A RACF group a UNIX group?
- Demonstration (via screen shots) of various shell activities
 - Seeing how you are defined to UNIX
 - Navigating the file system
 - Becoming a superuser
 - Displaying file attributes
 - Creating files and directories
 - Changing file permissions
 - Defining an access control list
- Goal: Start with a simple TSO user ID and get it working in the UNIX shell, right before your eyes, without skipping any steps



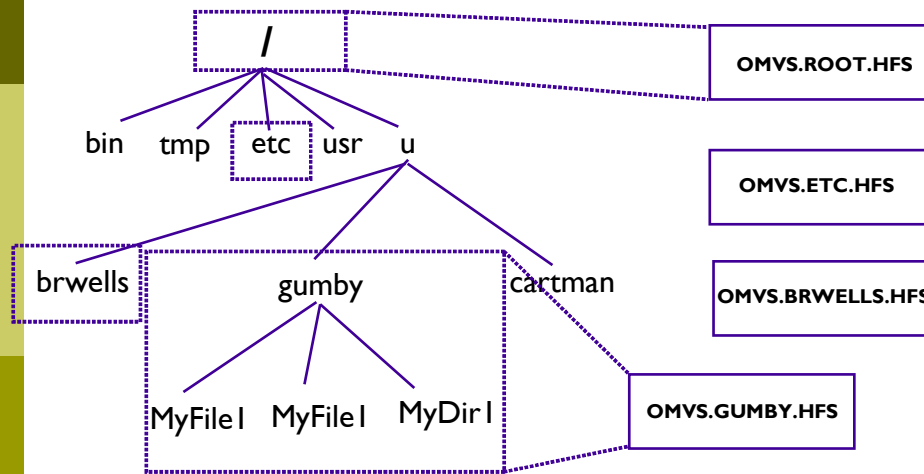
Background

- UNIX services first introduced as OpenEdition on MVS/SP V4.3
- Adhered to the IEEE POSIX standard, calling for:
 - A hierarchical file system
 - A set of APIs
 - An interactive shell environment with a defined minimum set of commands and utilities
- Concepts can be very foreign to z/OS users
- Perceived blurry line between what is the system programmer's responsibility and what is the security administrator's responsibility
 - Hint: It's always the sec admin's responsibility to secure z/OS
- Blurry line between what is owned/documented by RACF and what is owned/documented by z/OS UNIX



Brief overview of the UNIX file system

Data Sets are MOUNTed into a hierarchical structure



```
TSO MOUNT FILESYSTEM(OMVS.BRWELLS.HFS)
MOUNTPOINT('/u/brwells') MODE(RDWR) TYPE(HFS)
```

- Use the `df` (display file systems) shell command to see how it's all defined



Brief overview of the UNIX file system

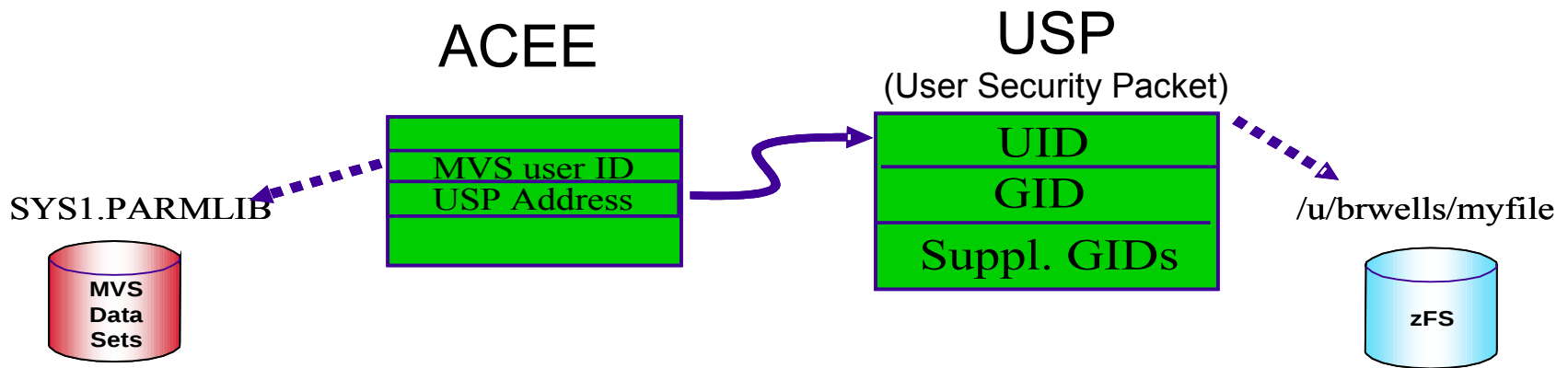
- All security information is kept with the file/directory as metadata in the file system.
 - Permission bits
 - Access control lists
 - Ownership
 - Audit settings
 - Extended attributes
- Kept in a SAF construct called the File Security Packet (FSP)
 - You can think of this as the “profile”
 - Contents displayed using the shell 'ls' command
 - Contents changed with a variety of shell commands, which we will see
- Several RACF classes control auditing with SETROPTS AUDIT and LOGOPTIONS



What makes a user a UNIX user?

- A UID!
 - The UID is an integer value uniquely (we hope) identifying a given user to the UNIX operating system (kernel)
 - It is used in all authorization decisions made by POSIX-compliant UNIX systems
 - It is thus used by z/OS UNIX, and mapped to a RACF user ID where necessary

- And a default group with a GID



How to make a UNIX user

1) Start with a normal RACF user ID

2) Fold in one OMVS segment with at minimum a UID

```
ALTUSER MYUSER OMVS(UID(1234567))
```

3) Sprinkle its default group with an OMVS segment with a GID

```
ALTGROUP MYGROUP OMVS(GID(7654321))
```



The user can now do UNIXy things

- Like entering the UNIX shell by typing the OMVS command in TSO

The screenshot shows a terminal window titled "E - POKVMTL4 - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window displays the following text:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

U.S. Government Users Restricted Rights -
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

FSUM2386 No shell program was specified in the user profile. The default shell
('/bin/sh') is used.
FSUM2383 No initial directory pathname was specified in the user profile. The h
ome directory is set to root.
$

===> _
INPUT
ESC=␣ 1=Help 2=SubCmd 3=HlpRetrn 4=Top 5=Bottom 6=TSO
7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MÁ E 21/007
Connected to remote server/host pokvmtl4.pok.ibr
  
```

- Note the defaults taken due to the sparse OMVS segment



Now what?

- The following set of shell commands should get you started

<u>Command</u>	<u>Purpose</u>
id	See how I am defined to UNIX
pwd	Print working directory name
cd	Change directory
ls	List files and directories, and their attributes
mkdir	Make a directory
oedit	Edit or create a file using ISPF
chmod	Change “file mode” (i.e. permission bits)
chown	Change file owner
find	Search for files with all sorts of attributes

- Consider printing them (especially ls) from the UNIX System Services Command Reference



Let's see who I am, where I am, and create a sandbox directory

```

F - POKVMTL4 - [24 x 80]
File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
GSA ADP Schedule Contract with IBM Corp.
IBM is a registered trademark of the IBM Corp.
FSUM2386 No shell program was specified in the user profile. The default shell
('/bin/sh') is used.
FSUM2383 No initial directory pathname was specified in the user profile. The h
ome directory is set to root.
$ # First off, note that you can enter comments to annotate your actions!
$ # And that the shell session is scrollable.
$ id
uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
$ pwd
/
$ ls
SYSTEM dev lib samples tmp usr
bin etc opt ssat u var
$ mkdir /u/myuser
mkdir: FSUM6404 directory "/u/myuser": EDC5111I Permission denied.
$
==> _
RUNNING
ESC=␣ 1=Help 2=SubCmd 3=HlpRetrn 4=Top 5=Bottom 6=TSO
7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MÁ F 21/007
Connected to remote serv
  
```

- Oops, I'm not authorized! Time to cheat...



Let's become a superuser with the su command

```

F - POKVMTL4 - [24 x 80]
File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
$ mkdir /u/myuser
mkdir: FSUM6404 directory "/u/myuser": EDC5111I Permission denied.
$ # Now i will attempt to enter superuser mode using the su command
$ su
FSUM5011 su: User not authorized to obtain superuser authority.
$ I have SPECIAL, so I will now permit myself. From within the shell!
I: FSUM7351 not found
$ tsocmd "PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(MYUSER) ACCESS(READ)"
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(MYUSER) ACCESS(READ)
RACLISTED PROFILES FOR FACILITY WILL NOT REFLECT THE UPDATE(S) UNTIL A SETROPTS
REFRESH IS ISSUED
$ tsocmd "SETROPTS RACLIST(FACILITY) REFRESH"
SETROPTS RACLIST(FACILITY) REFRESH
$ id
uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
$ su
# id
uid=0(HZSPROC) gid=7654321(MYGROUP) groups=1(SYS1)
# # And note the change of the prompt character to "#"
#
===> _
RUNNING
ESC=⌘ 1=Help 2=SubCmd 3=HlpRetrn 4=Top 5=Bottom 6=TSO
7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MA F 21/007
Connected to remote serv

```

Now we can create our directory and exit from superuser mode

The screenshot shows a terminal window titled "F - POKVMTL4 - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal content is as follows:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
SETROPTS RACLIST(FACILITY) REFRESH
$ id
uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
$ su
# id
uid=0(HZSPROC) gid=7654321(MYGROUP) groups=1(SYS1)
# # And note the change of the prompt character to "#"
# mkdir /u/myuser
# ls -ld /u/myuser
drwxr-xr-x  2 HZSPROC  SYS1      8192 Nov 20 15:24 /u/myuser
# # Since I created it as superuser, it is owned by superuser.
# # So change the owner to MYUSER
# chown myuser /u/myuser
# ls -ld /u/myuser
drwxr-xr-x  2 MYUSER   SYS1      8192 Nov 20 15:24 /u/myuser
# exit
$ # Now I'm MYUSER again
$ cd /u/myuser
$ ls
$
===> _
RUNNING
ESC=⌘  1=Help    2=SubCmd   3=HlpRetrn 4=Top      5=Bottom   6=TSO
        7=BackScr  8=Scroll   9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MA F 21/007
Connected to remote serv

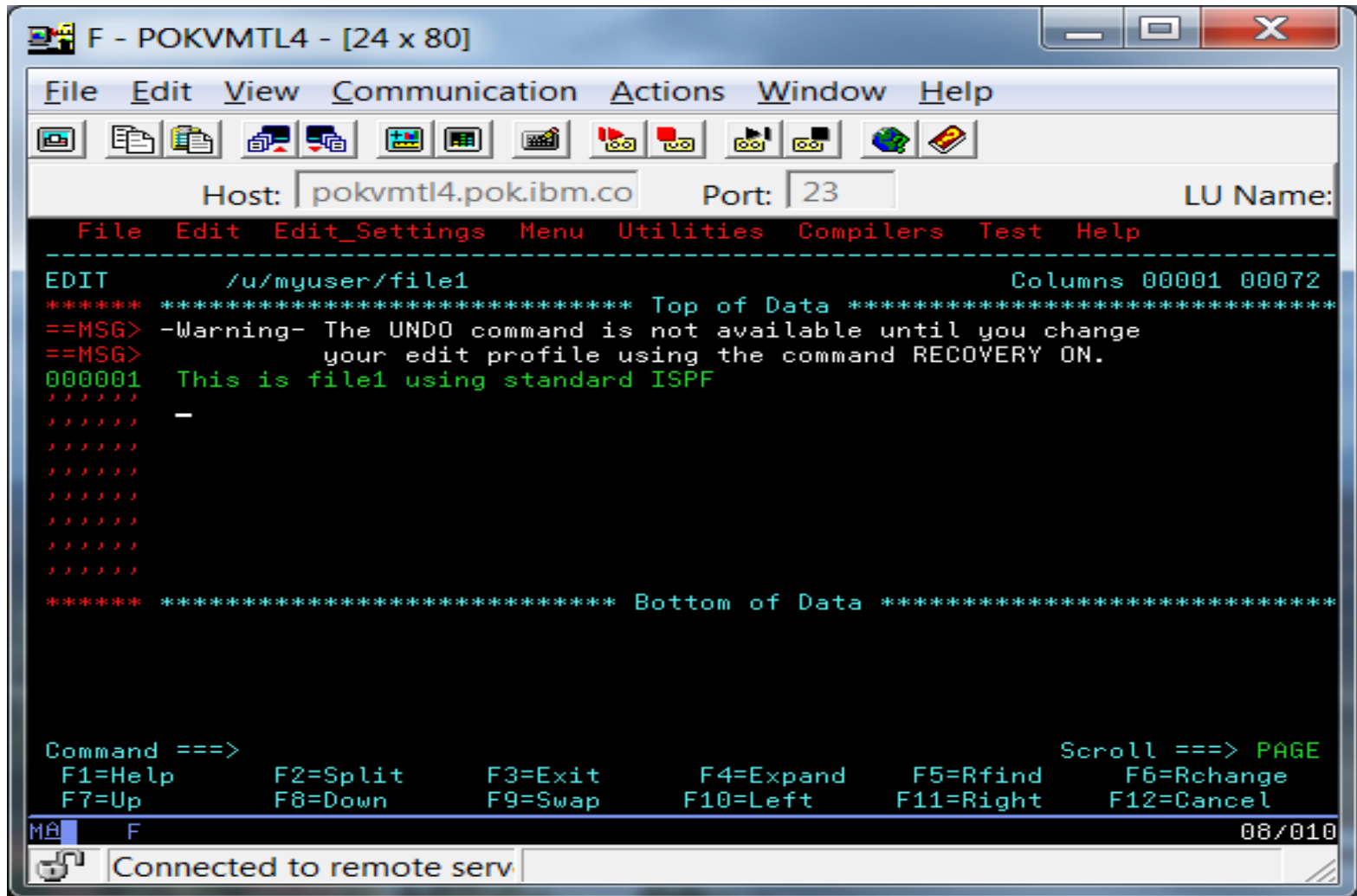
```

And we can create files in our new directory

```

F - POKVMTL4 - [24 x 80]
File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
$ su
# id
uid=0(HZSPROC) gid=7654321(MYGROUP) groups=1(SYS1)
# # And note the change of the prompt character to "#"
# mkdir /u/myuser
# ls -ld /u/myuser
drwxr-xr-x  2 HZSPROC  SYS1          8192 Nov 20 15:24 /u/myuser
# # Since I created it as superuser, it is owned by superuser.
# # So change the owner to MYUSER
# chown myuser /u/myuser
# ls -ld /u/myuser
drwxr-xr-x  2 MYUSER  SYS1          8192 Nov 20 15:24 /u/myuser
# exit
$ # Now I'm MYUSER again
$ cd /u/myuser
$ ls
$ # Let's see 3 ways of creating a file
$ # First, use oedit to create one with ISPF edit
$
==> oedit file1_
INPUT
ESC=␣  1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll   9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
Mâ F 21/018
Connected to remote serv
  
```


Using good old ISPF



And now using a couple of UNIX techniques

```

F - POKVMTL4 - [24 x 80]
File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
# ls -ld /u/myuser
drwxr-xr-x  2 MYUSER  SYS1      8192 Nov 20 15:24 /u/myuser
# exit
$ # Now I'm MYUSER again
$ cd /u/myuser
$ ls
$ # Let's see 3 ways of creating a file
$ # First, use oedit to create one with ISPF edit
$ oedit file1
$ # Second, the UNIX touch command
$ touch file2
$ # Third, use the shell redirection operator to put cmd output into a file
$ id > file3
$ ls -l
total 16
-rwx-----  1 MYUSER  SYS1      35 Nov 20 15:30 file1
-rw-r--r--   1 MYUSER  SYS1       0 Nov 20 15:30 file2
-rw-r--r--   1 MYUSER  SYS1     56 Nov 20 15:31 file3
$ # Note that different commands can set different initial permissions
$
==> _
RUNNING
ESC=⋄  1=Help    2=SubCmd   3=HlpRetrn 4=Top      5=Bottom   6=TSO
        7=BackScr 8=Scroll   9=NextSess 10=Refresh 11=FwdRetr 12=Retrieve
MA F 21/007
Connected to remote serv

```

–And list their attributes using the ls command

UNIX file concepts

```
-rwx----- 1 MYUSER  SYS1      25 Oct 17 09:36 file1
-rw-r--r--  1 MYUSER  SYS1      0 Oct 17 09:37 file2
-rw-r--r--  1 MYUSER  SYS1     56 Oct 17 09:39 file3
```

- Files have an owner (MYUSER), and also have a group owner (SYS1)!
 - The group owner defaults to that of the parent directory
 - In reality, it's the numeric UID and GID that are stored in the file
 - The ls command is mapping them to user IDs and group names for your convenience (use -n option to see numeric values)
- Files can be accessed 3 ways: read (r), write (w), and execute (x)
 - Unlike RACF profile access levels, these are not hierarchical
- Each file has three sets of permission bits
 - The **left-most set** applies to the file's owner
 - The **middle set** applies to the file's group owner
 - The **right-most set** applies to everyone else (think UACC)



Where did those initial permissions come from?

- Each command, utility, operator, etc specifies initial permission on the open() API call they make to create the file.
 - Many POSIX-compliant commands create files with overly permissive access
 - The non-POSIX z/OS extensions, for example oedit and OPUT, create files with more secure defaults

- The umask (user mask) can be used to mask off undesirable initial permissions at file creation time (not at chmod time).
 - Often used to prevent “world-write” sneaking in on you
 - Often configured in /etc/profile (The system-wide profile for z/OS shell users. It contains environment variables and commands used by most shell users.)



Demonstrating umask

The screenshot shows a terminal window titled "F - POKVMTL4 - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal content is as follows:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
$ umask
0022
$ # That was octal notation. It indicates the bits that are masked OFF.
$ # Symbolic notation is more intuitive, and is the inverse of octal.
$ # Using symbolic notation, you are shown the bits ALLOWED to be set
$ umask -S
u=rwx,g=rx,o=rx
$ # Let's reset umask so that no bits are masked off
$ umask 0
$ # Now display it again
$ umask
0000
$ # Now create another file
$ touch file4
$ ls -l
total 16
-r-x-----  1 MYUSER  SYS1      10 Nov 20 15:42 file1
-rw-r--r--  1 MYUSER  SYS1       0 Nov 20 15:30 file2
-rw-r--r--  1 MYUSER  SYS1     56 Nov 20 15:31 file3
-rw-rw-rw-  1 MYUSER  SYS1       0 Nov 20 15:49 file4
===> _

```

At the bottom of the terminal, there is a status bar with the text "Connected to remote serv" and a cursor. The bottom right corner of the terminal shows "21/007".

Who controls umask?

- The file owner has complete control of her umask value
 - Can be configured in .profile file in user's home directory, which would override the /etc/profile setting
 - The umask command can be issued at any time
 - There are no security controls over umask
 - And regardless of umask, the file owner can always change permissions



Changing permissions with the chmod command

- chmod – change file mode
 - The permissions exist in a bit-string called the file mode
- chmod can be used with octal or symbolic notation
 - Symbolic tends to be more intuitive

- Let's take away our own access to demonstrate
 - We will use the cat (con-cat-enate) command to display file contents
 - The head and tail commands are also useful, especially when the file is large



Remove our read access

The screenshot shows a terminal window titled "F - POKVMTL4 - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal displays the following commands and output:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
$ touch file2
$ # Third, use the shell redirection operator to put cmd output into a file
$ id > file3
$ ls -l
total 16
-rwx----- 1 MYUSER  SYS1      35 Nov 20 15:30 file1
-rw-r--r--  1 MYUSER  SYS1      0 Nov 20 15:30 file2
-rw-r--r--  1 MYUSER  SYS1     56 Nov 20 15:31 file3
$ # Note that different commands can set different initial permissions
$ cat file1
This is file1 using standard ISPF
$ # Let's take away the file owner's read access
$ chmod u-r file1
$ ls -l file1
--wx----- 1 MYUSER  SYS1      35 Nov 20 15:30 file1
$ cat file1
cat: file1: EDC5111I Permission denied.
$ # Since permissions are not hierarchical, we can still write to it
$ echo more text > file1
$
===> _

```

At the bottom of the terminal window, there is a status bar with the text "Connected to remote serv" and a "RUNNING" indicator. The status bar also shows a cursor and the text "21/007".

Remove our write access

```

File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
-rw-r--r--  1 MYUSER  SYS1      0 Nov 20 15:30 file2
-rw-r--r--  1 MYUSER  SYS1      56 Nov 20 15:31 file3
$ # Note that different commands can set different initial permissions
$ cat file1
This is file1 using standard ISPF
$ # Let's take away the file owner's read access
$ chmod u-r file1
$ ls -l file1
--wx-----  1 MYUSER  SYS1      35 Nov 20 15:30 file1
$ cat file1
cat: file1: EDC5111I Permission denied.
$ # Since permissions are not hierarchical, we can still write to it
$ echo more text > file1
$ chmod u+r file1
$ cat file1
more text
$ chmod u-w file1
$ echo now write this > file1
FSUM7343 cannot open "file1" for output: EDC5111I Permission denied.
$
===> _
RUNNING
ESC=⌘  1=Help    2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
       7=BackScr  8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA F 21/007
Connected to remote serv

```

We can augment permission bits with access control lists

- Created using `setfacl`, listed using `getfacl`
- Acls are an extension to the POSIX standard
- Can be defined/managed at any time, but require the FSSEC class to be active before they are used in access decisions
- Permissions **and** acls are used in determining file access



Create and list an acl

The screenshot shows a terminal window titled "F - POKVMTL4 - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu bar is a toolbar with various icons. The terminal content is as follows:

```

Host: pokvmtl4.pok.ibm.co      Port: 23      LU Name:
-rw-rw-rw-  1 MYUSER  SYS1      0 Nov 20 15:49 file4
$ setfacl -m u:tsousr4:r--,g:sys1:r-x file4
$ ls -l file4
-rw-rw-rw+  1 MYUSER  SYS1      0 Nov 20 15:49 file4
$ # Now, a plus sign indicates an acl exists. Use getfacl to show it.
$ getfacl file4
#file: file4
#owner: MYUSER
#group: SYS1
user::rw-
group::rw-
other::rw-
user:TSOUSR4:r--
group:SYS1:r-x

$ # getfacl also shows ownership and permissions
$ # This gives you a comprehensive view of file access
$

===> _
RUNNING
ESC=⌘  1=Help      2=SubCmd      3=HlpRetrn   4=Top        5=Bottom     6=TSO
        7=BackScr    8=Scroll     9=NextSess  10=Refresh   11=FwdRetr   12=Retrieve
Mâ F 21/007
Connected to remote serv

```

What other cool security information can we see using ls?

- Audit bits with the `-W` option

```
-rwx-----  fff---  1 MYUSER  SYS1          25 Oct 17 09:36 file1
-rw-r--r--  fff---  1 MYUSER  SYS1           0 Oct 17 09:37 file2
-rw-r--r--  fff---  1 MYUSER  SYS1          56 Oct 17 09:39 file3
```

- Each file has two sets of audit bits

- The **left set** contains the file owner's options
- The **right set** contains the system AUDITOR's options

- The owner options are initialized to log failed (“f”) access for read, write, and execute. The AUDITOR options are off.
- Note the parallel with AUDIT and GLOBALAUDIT, and their default settings, in a RACF general resource or DATASET profile!!!
- Audit settings are managed with the `chaudit` command
- Audit bits are an extension to the POSIX standard



What other cool security information can we see using ls?

- Extended attribute bits with the -E option

```
-rwx-----  --s-  1 MYUSER  SYS1          25 Oct 17 09:36 file1
-rw-r--r--  --s-  1 MYUSER  SYS1           0 Oct 17 09:37 file2
-rw-r--r--  -s-  1 MYUSER  SYS1          56 Oct 17 09:39 file3
```

- There are four possible extended attributes, all applicable only to executable files

- The **first bit** indicates whether the file is considered APF authorized
- The **second bit** indicates whether the file is considered to be program-controlled
- The **third bit** indicates whether the file is enabled to run in a shared address space
- The **fourth bit** indicates whether the file is loaded from the shared library region

- Extended attribute settings are managed with the extattr command

- Extended attributes are an extension to the POSIX standard



Bonus tip of the day – save your shell session to a file!

- Discovered by yours truly just after the nick of time

The screenshot shows a terminal window titled 'F - POKVMTL4 - [24 x 80]'. The window has a menu bar (File, Edit, View, Communication, Actions, Window, Help) and a toolbar. The terminal content is as follows:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
('/bin/sh') is used.
FSUM2383 No initial directory pathname was specified in the user profile. The h
ome directory is set to root.
$ script /u/myuser/SaveMyShellSession
Script command is started. The file is /u/myuser/SaveMyShellSession.
$ ls /u/myuser
SaveMyShellSession  file2          file4
file1               file3
$ tsocmd listgrp mygroup omvs noracf
listgrp mygroup omvs noracf
INFORMATION FOR GROUP MYGROUP

OMVS INFORMATION
-----
GID= 0007654321
$ id
uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
$ exit
Script command is complete. The file is /u/myuser/SaveMyShellSession.
$
===> oedit /u/myuser/SaveMyShellSession

```

At the bottom of the terminal, there is a status bar with the text 'Connected to remote serv' and a date '21/013'.

The script command was introduced in z/OS V1R13



The fruits of our labor captured for posterity

```

F - POKVMTL4 - [24 x 80]
File Edit View Communication Actions Window Help
Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT /u/myuser/SaveMyShellSession Columns 00001 00072
***** ***** Top of Data *****
000001 Script command is started on Fri Nov 21 09:11:19 2014.
000002 $ ls /u/myuser
000003 SaveMyShellSession file2 file4
000004 file1 file3
000005 $ tsocmd listgrp mygroup omvs noracf
000006 listgrp mygroup omvs noracf
000007 INFORMATION FOR GROUP MYGROUP
000008
000009 OMVS INFORMATION
000010 -----
000011 GID= 0007654321
000012 $ id
000013 uid=1234567(MYUSER) gid=7654321(MYGROUP) groups=1(SYS1)
000014 $ exit
000015 Script command is complete on Fri Nov 21 09:11:55 2014.
***** ***** Bottom of Data *****

Command ==>
F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
Scroll ==> PAGE
MA F 22/015
Connected to remote serv
  
```



Bonus tip #2 – accessing MVS data sets from the shell!

The screenshot shows a terminal window titled "F - POKVMTL4 - [24 x 80]". The window has a menu bar (File, Edit, View, Communication, Actions, Window, Help) and a toolbar. The terminal displays the following commands and output:

```

Host: pokvmtl4.pok.ibm.co Port: 23 LU Name:
$ head "'SYS1.PARMLIB.POK(BPXPRMA4)'"
LIMMSG(ALL)
MAXFILEPROC(200)
MAXPROCSYS(1000)
MAXPROCUSER(50)
MAXCPUPTIME(2147483647)
MAXUIDS(200)
MAXPTYS(256)
CTRACE(CTIBPX00)
STEPLIBLIST('/system/steplib')
FILESYSTYPE TYPE(ZFS)
$ cp "'SYS1.PARMLIB.POK(BPXPRMA4)'" MyBpxParms
$ tail -5 MyBpxParms
        ENTRYPOINT(EZBPFINI)
        DEFAULT
NETWORK DOMAINNAME(AF_INET) DOMAINNUMBER(2) MAXSOCKETS(10000)
        TYPE(CINET) INADDRANYPORT(6000) INADDRANYCOUNT(1000)
RESOLVER_PROC(RESOLVER)
$
===> _
INPUT
ESC=␣  1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll   9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA F 21/007
Connected to remote serv
  
```

Oh, and about that original mkdir violation ...



- Let's look at the console to see the result of that violation

```
00  SYS1  ICH408I  USER(MYUSER  )  GROUP(MYGROUP  )  NAME(#####)
      /u/myuser  CL(DIRACC  )  FID(01C8D9E9F2F2F0000104000000190000)
      INSUFFICIENT  AUTHORITY  TO  MKDIR
      ACCESS  INTENT(-W-)  ACCESS  ALLOWED(GROUP          R-X)
      EFFECTIVE  UID(0001234567)  EFFECTIVE  GID(0007654321)
```

- UNIX file and directory violations are logged by default, due to
 - the DIRSRCH, DIRACC and FSOBJ classes being in the SETROPTS LOGOPTIONS(DEFAULT) list
 - The fact that files and directories have audit settings that are defaulted to log violations (just like RACF profiles!)
- The ICH408I message is keyed off of the creation of the SMF record



Advanced topics for your future research

- File system mount modes and BPXPRMxx member of PARMLIB
- FSACCESS class to control access to file system data sets
- Organizing the file system data sets for granular security control
- UNIX user-provisioning with BPX.UNIQUE.USER and automount
- Using the find command and the irrhfsu (HFS Unload) utility to analyze file attributes
- Superuser authority and the various ways of granting it
- Daemons/servers, identity switches; their dangers and controls
- acl inheritance



Good sources of information: z/OS UNIX

- UNIX System Services web site
 - <http://www-1.ibm.com/servers/eserver/zseries/zos/unix/>
 - Check out the Tools page under the Tips tab
- UNIX System Services Planning manual
 - Especially the security chapter
- UNIX System Services Command Reference
- mvs-oe mailing list
 - see the “Where to find more information” section in the front matter of any z/OS UNIX publication



Good sources of information: RACF

- RACF web site
 - <http://www-1.ibm.com/servers/eserver/zseries/zos/racf/>
 - See the Downloads page under the Resources tab for HFS Unload
- RACF Security Administrator's Guide (UNIX chapter)
- RACF Auditor's Guide
- racf-l mailing list
 - see the “Other sources of information” section in the front matter of any RACF publication



