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.

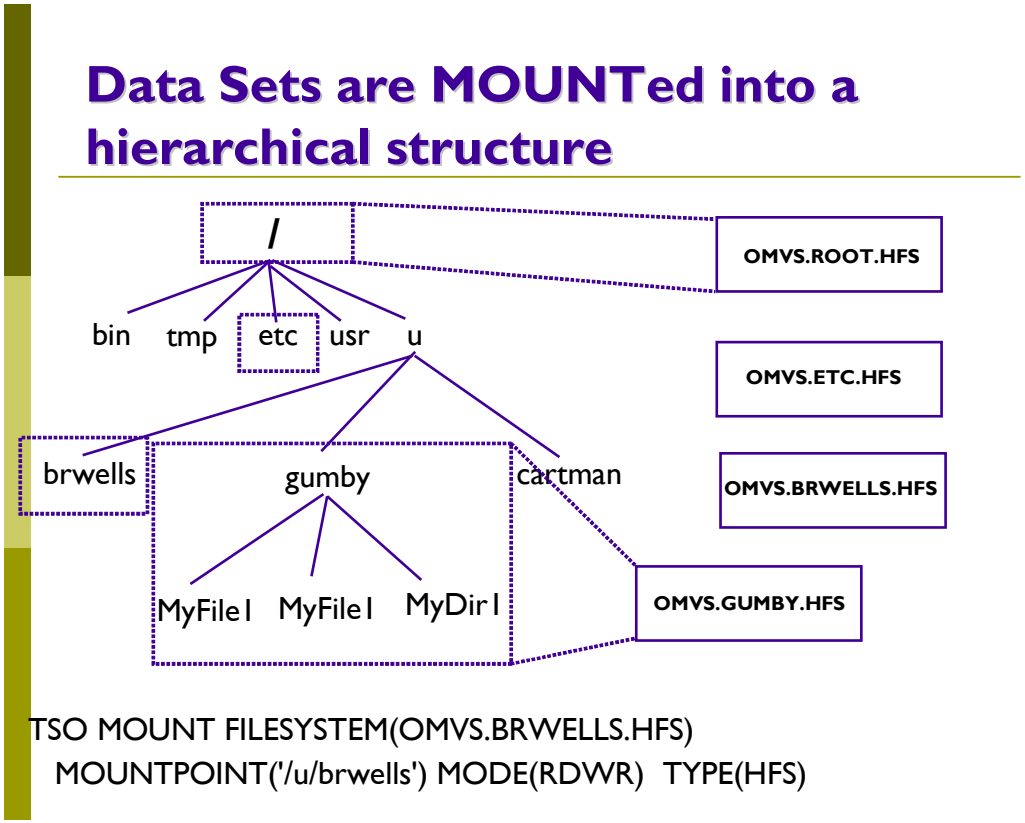
Agenda

- 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



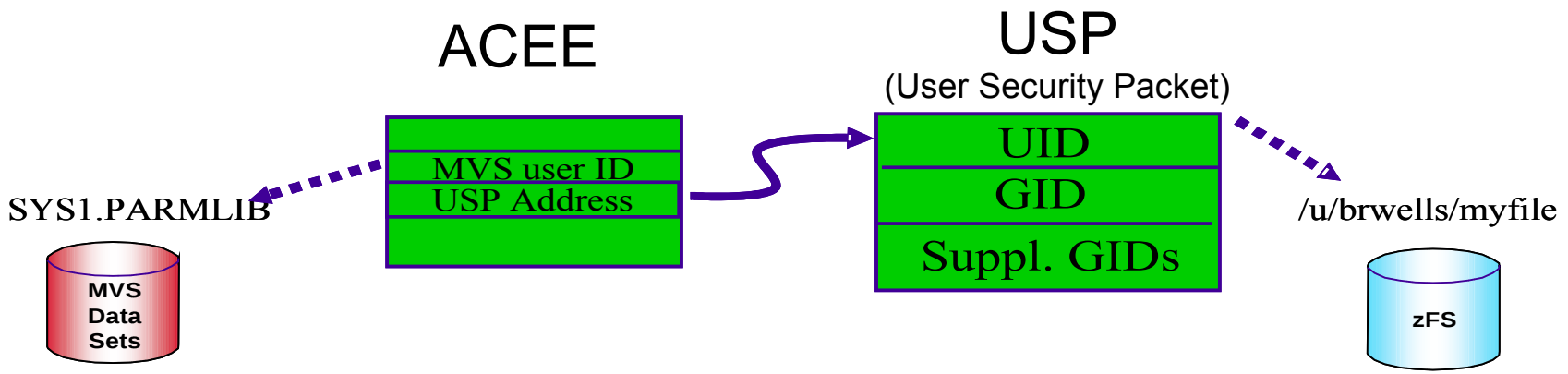
- Use the df (display file systems) shell command to see how it's all defined

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

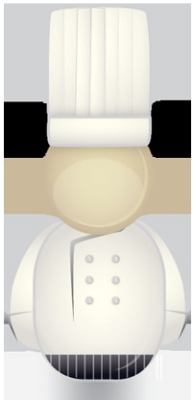
1) Start with a normal RACF user ID

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

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

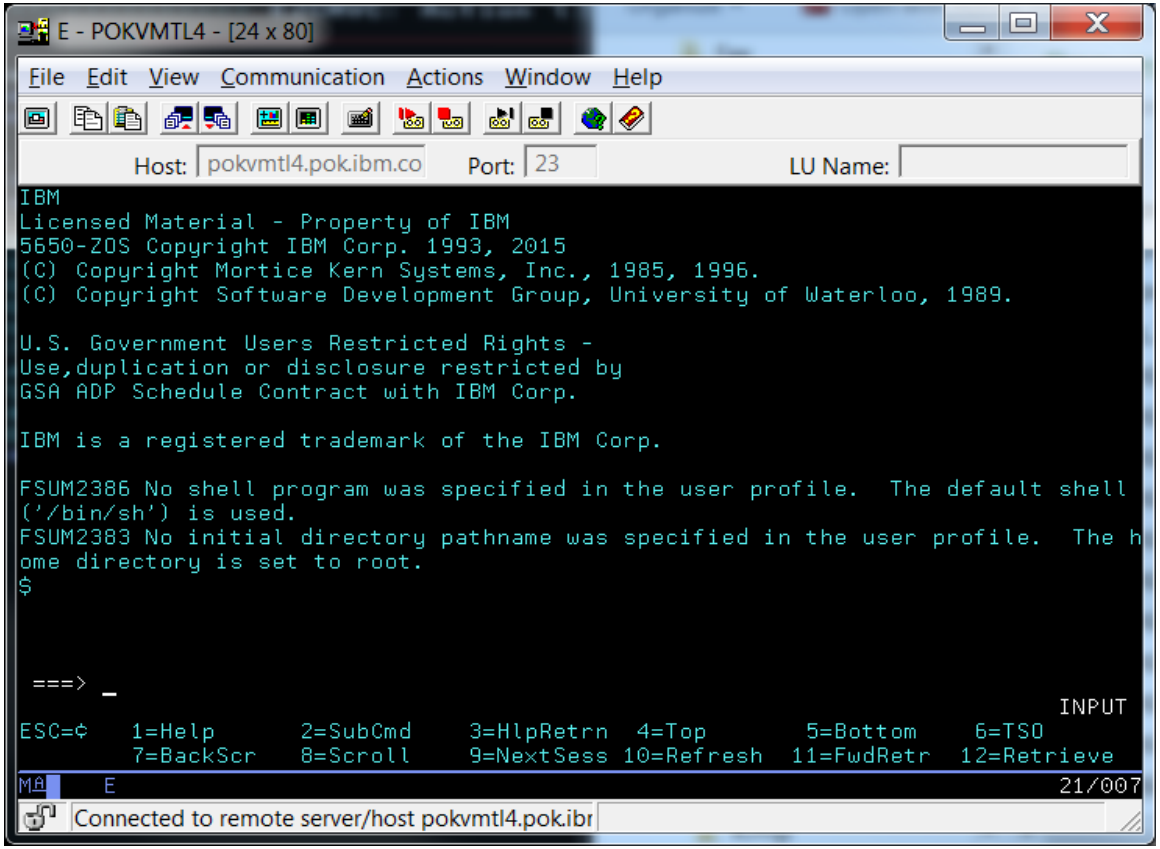
3) Sprinkle its default group with an OMVS segment with a GID

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



The user can now do UNIXy things

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



- Note the defaults taken due to the sparse OMVS segment

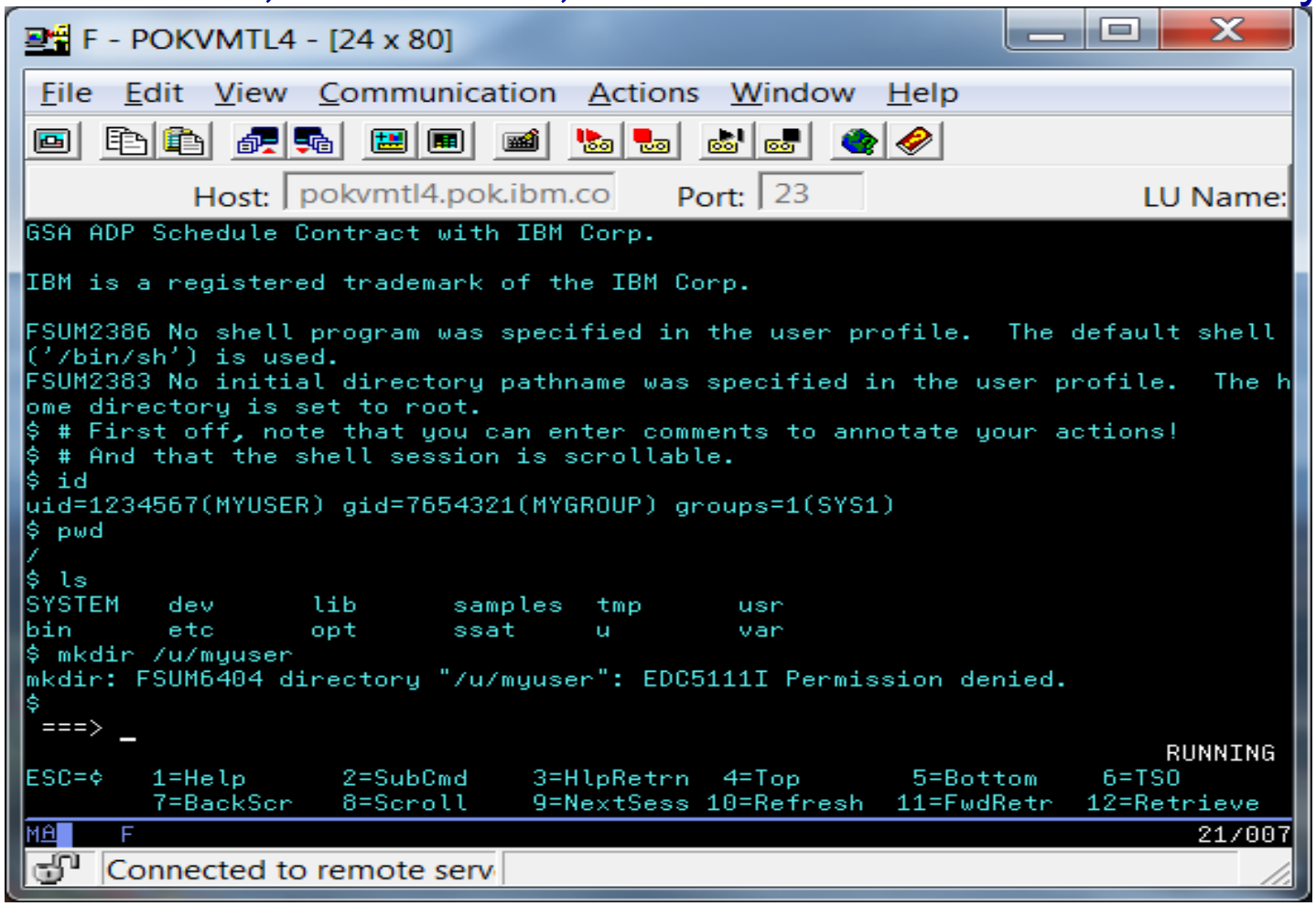
Now what?

- The following set of shell commands should get you started

Command	Purpose
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

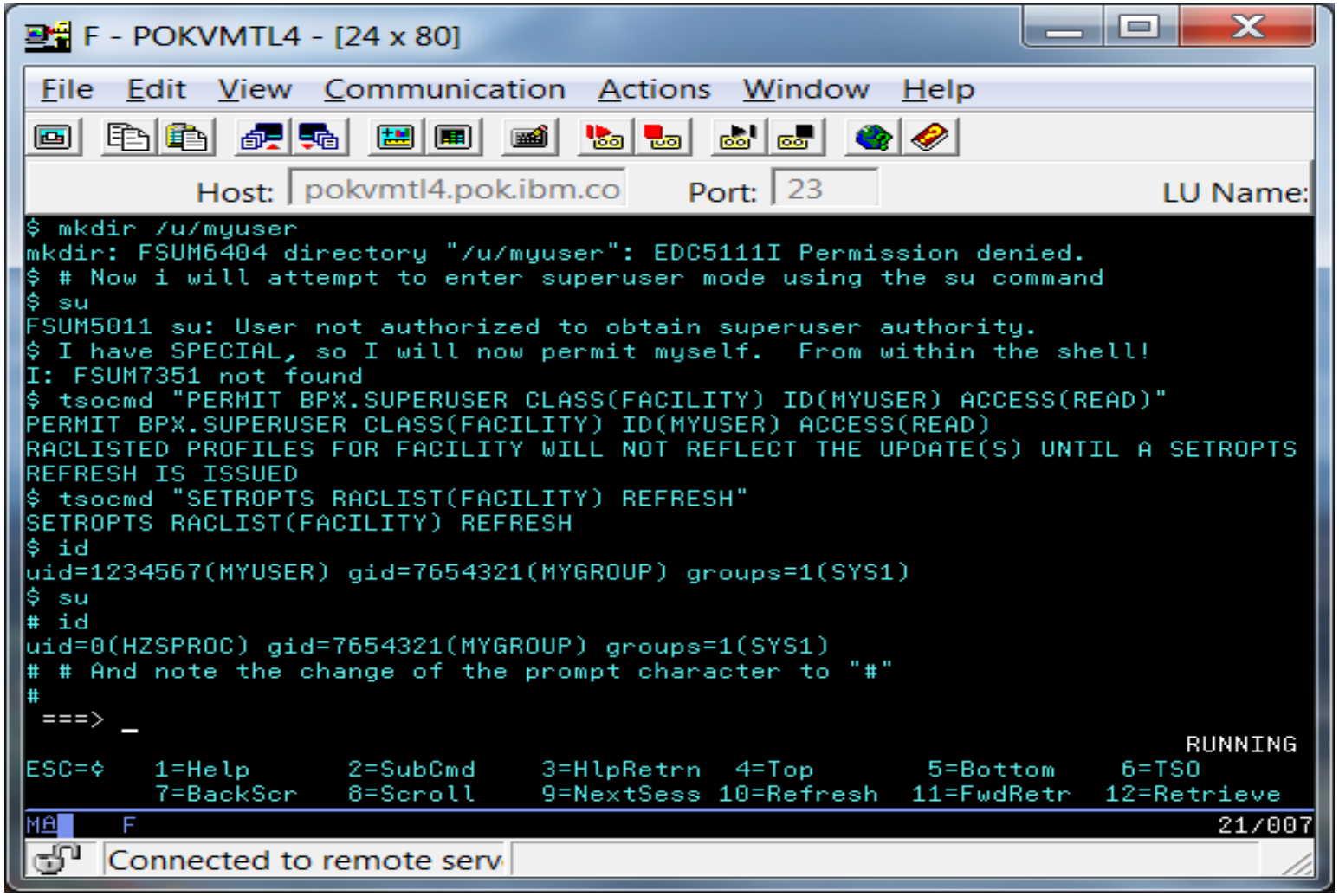
- Consider printing them (especially ls) from the UNIX System Services Command Reference

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

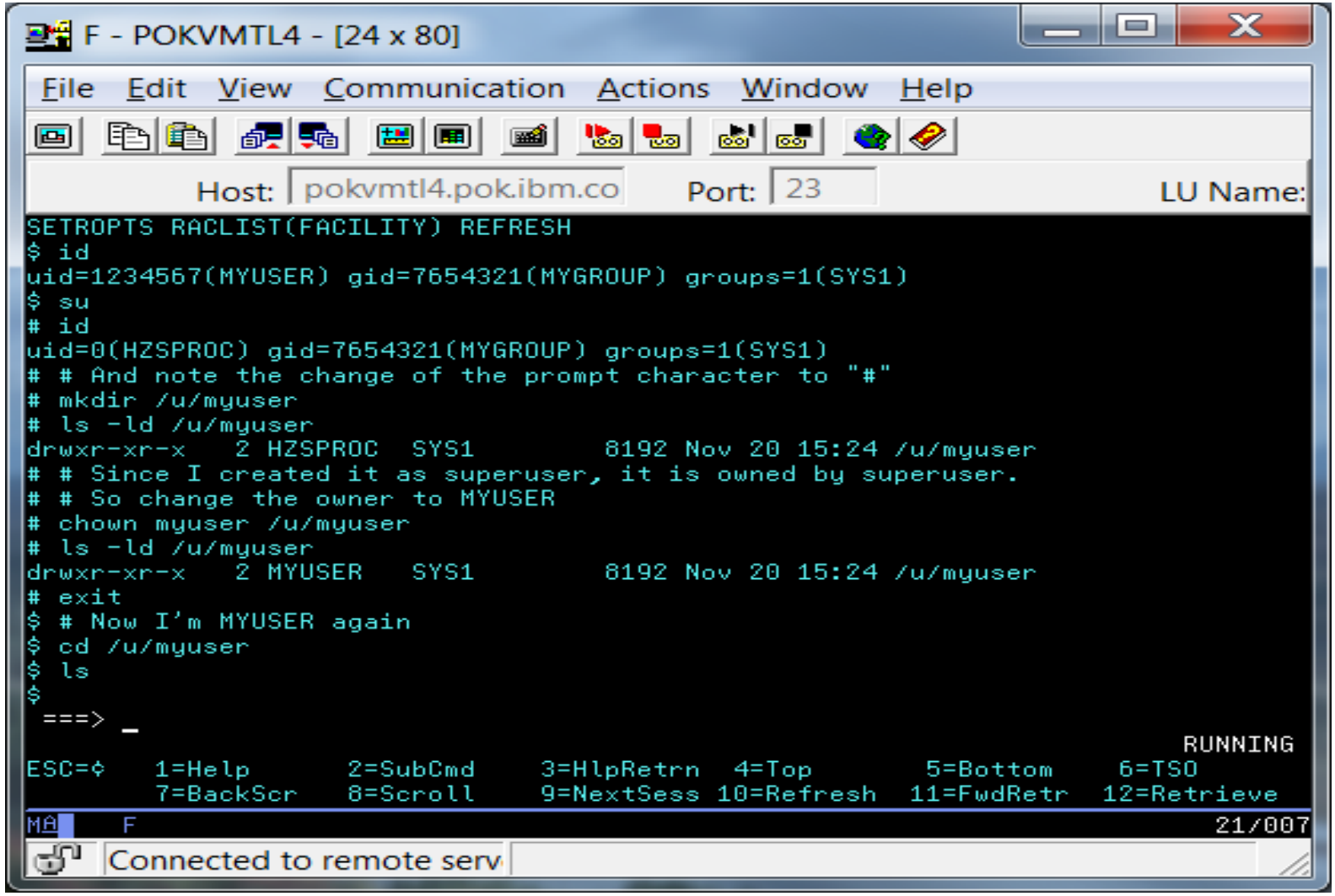


- Oops, I'm not authorized! Time to cheat...

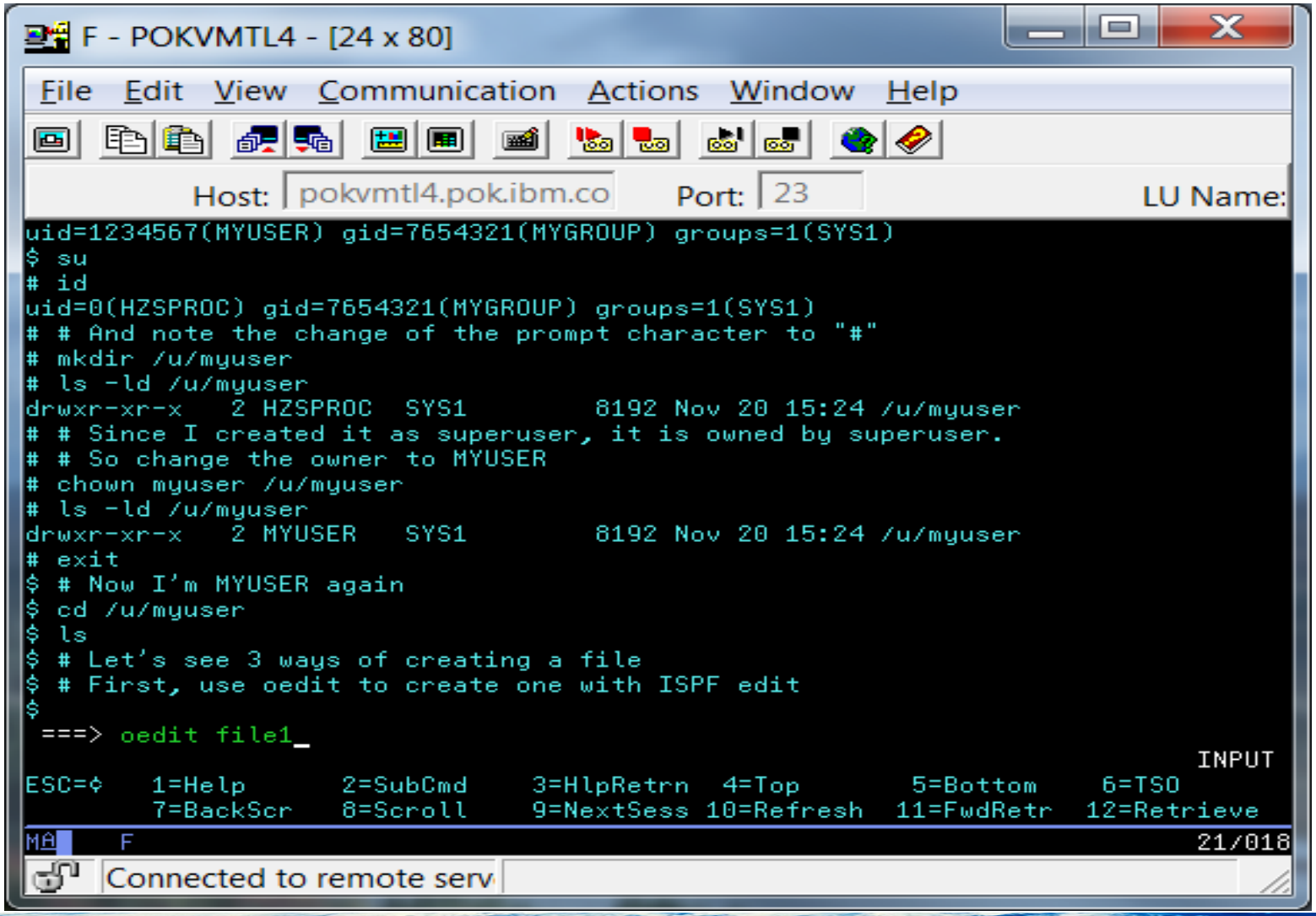
Let's become a superuser with the su command



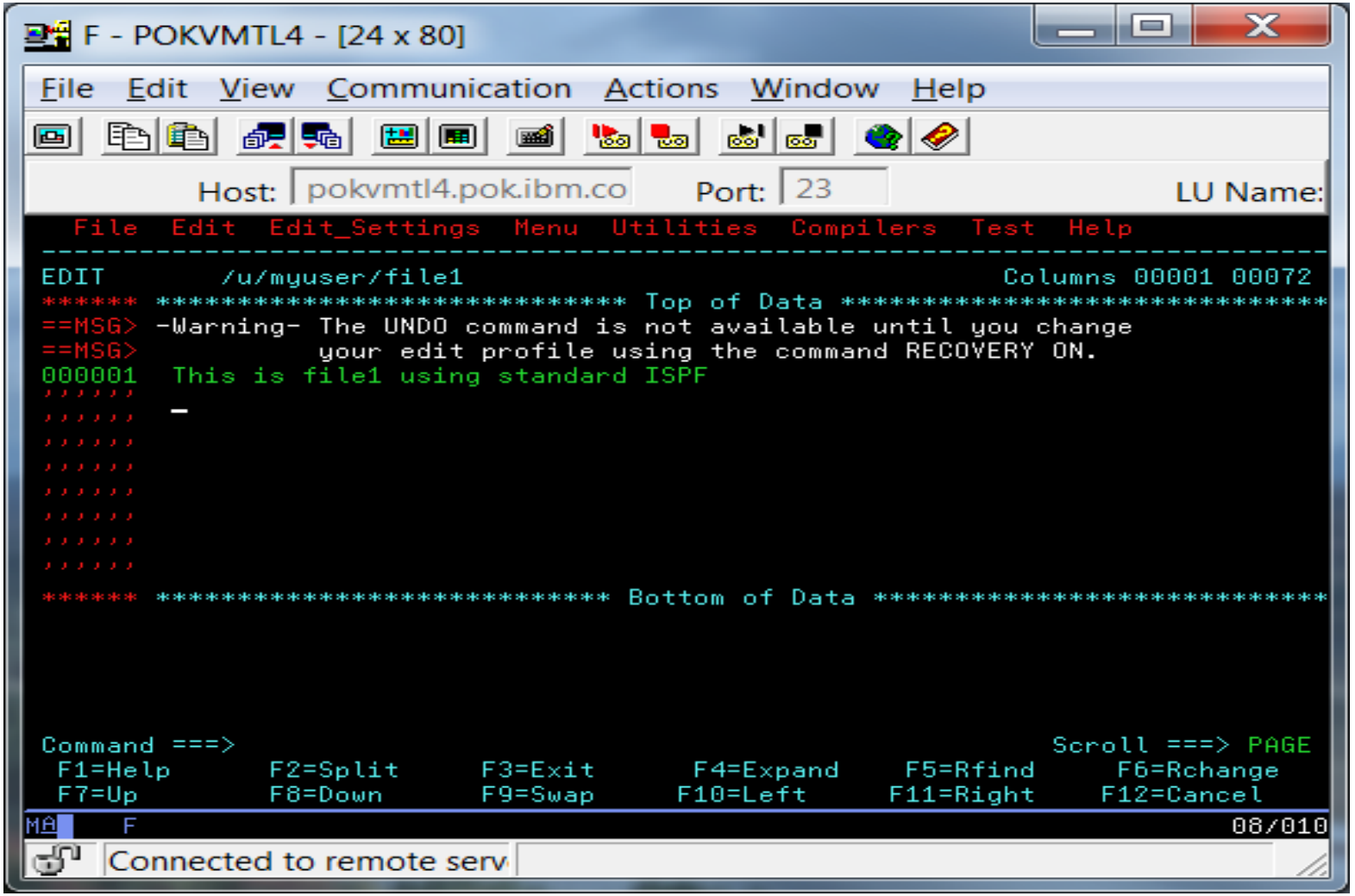
Now we can create our directory and exit from superuser mode



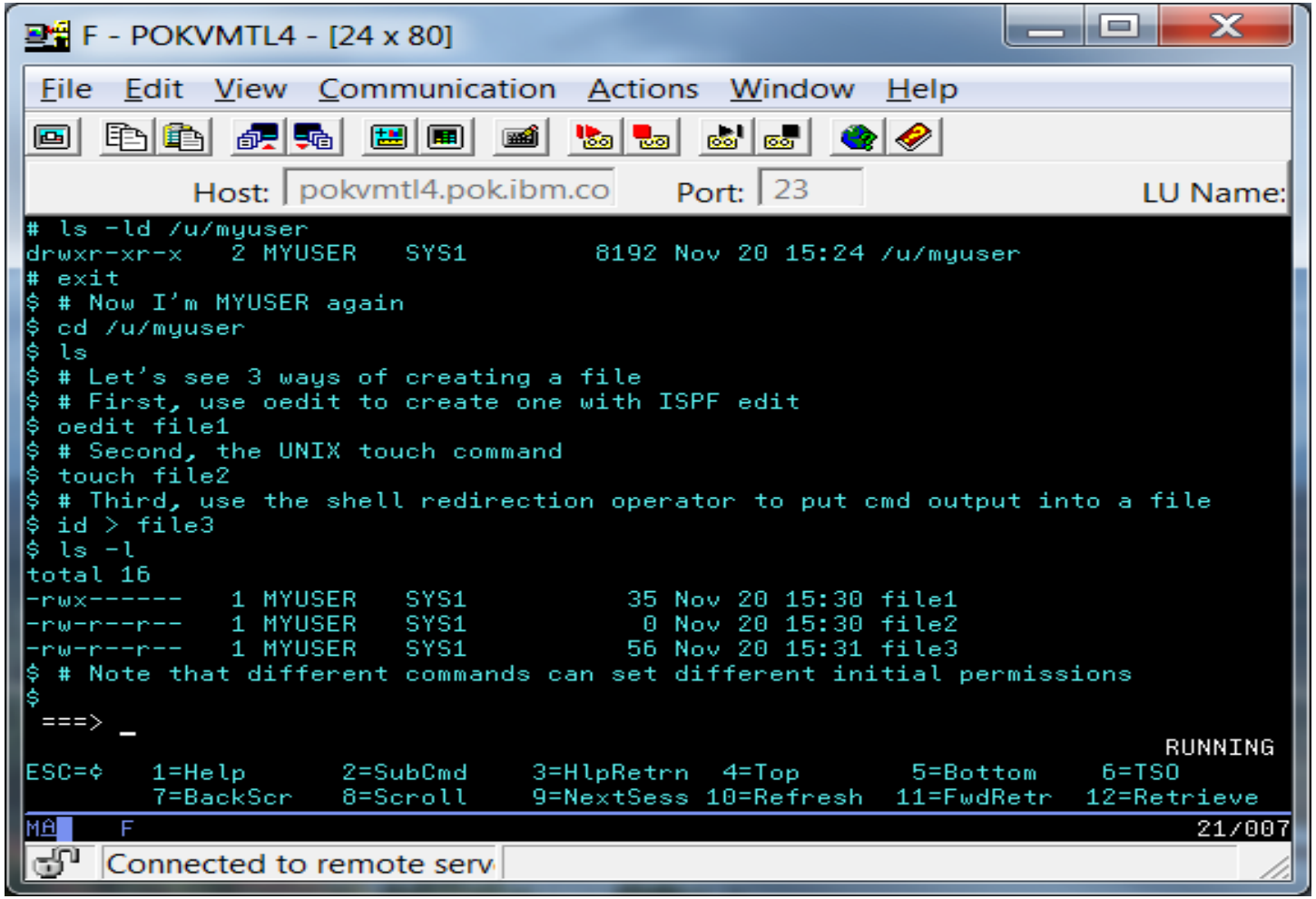
And we can create files in our new directory



Using good old ISPF



And now using a couple of UNIX techniques



– 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
```

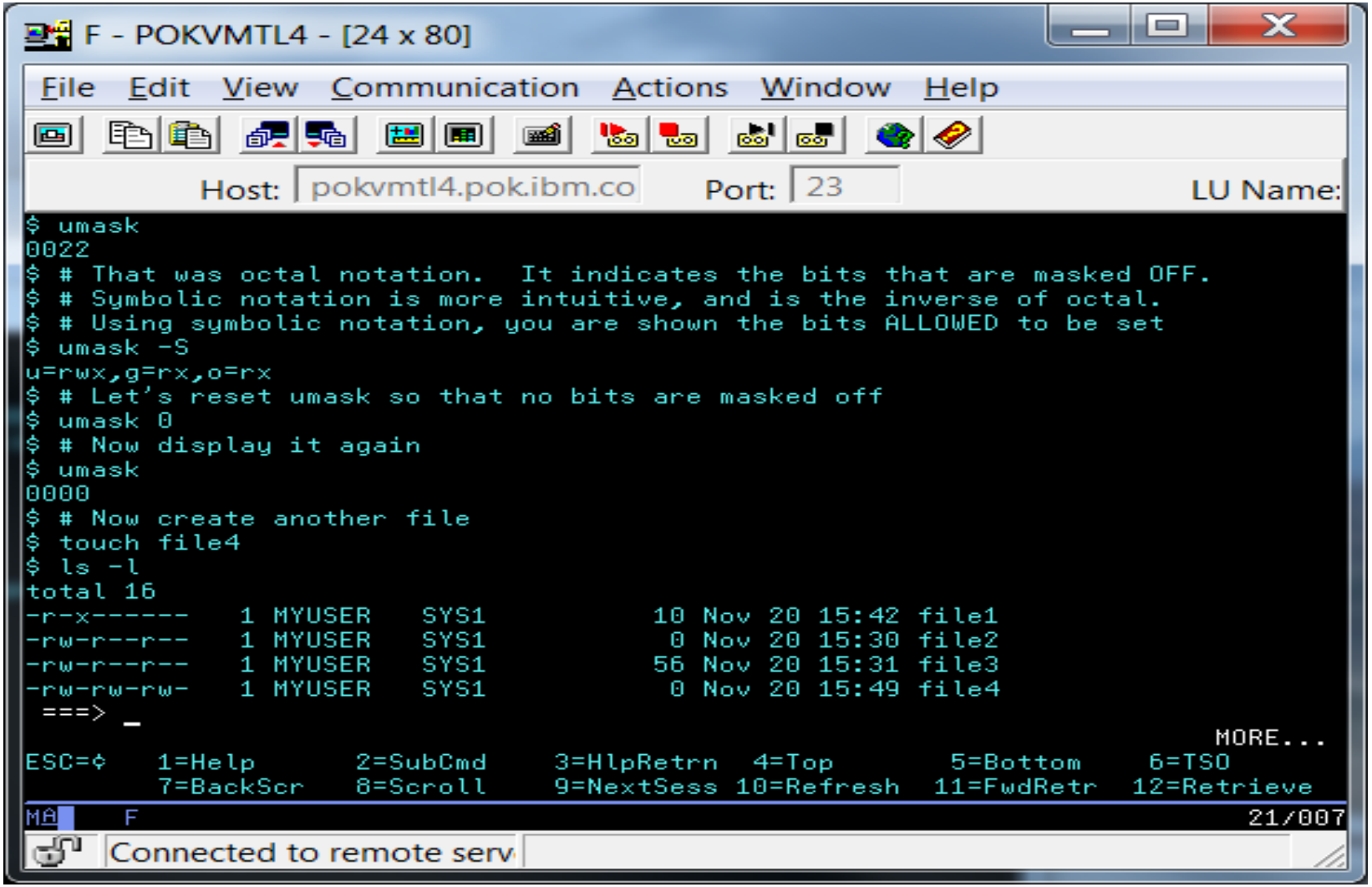
- Files have an owner (MYUSER), and also have a group owner (SYS1)!
  - The group owner defaults to that of the parent directory
  - In reality, it's the numeric UID and GID that are stored in the file
    - The ls command is mapping them to user IDs and group names for your convenience (use -n option 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



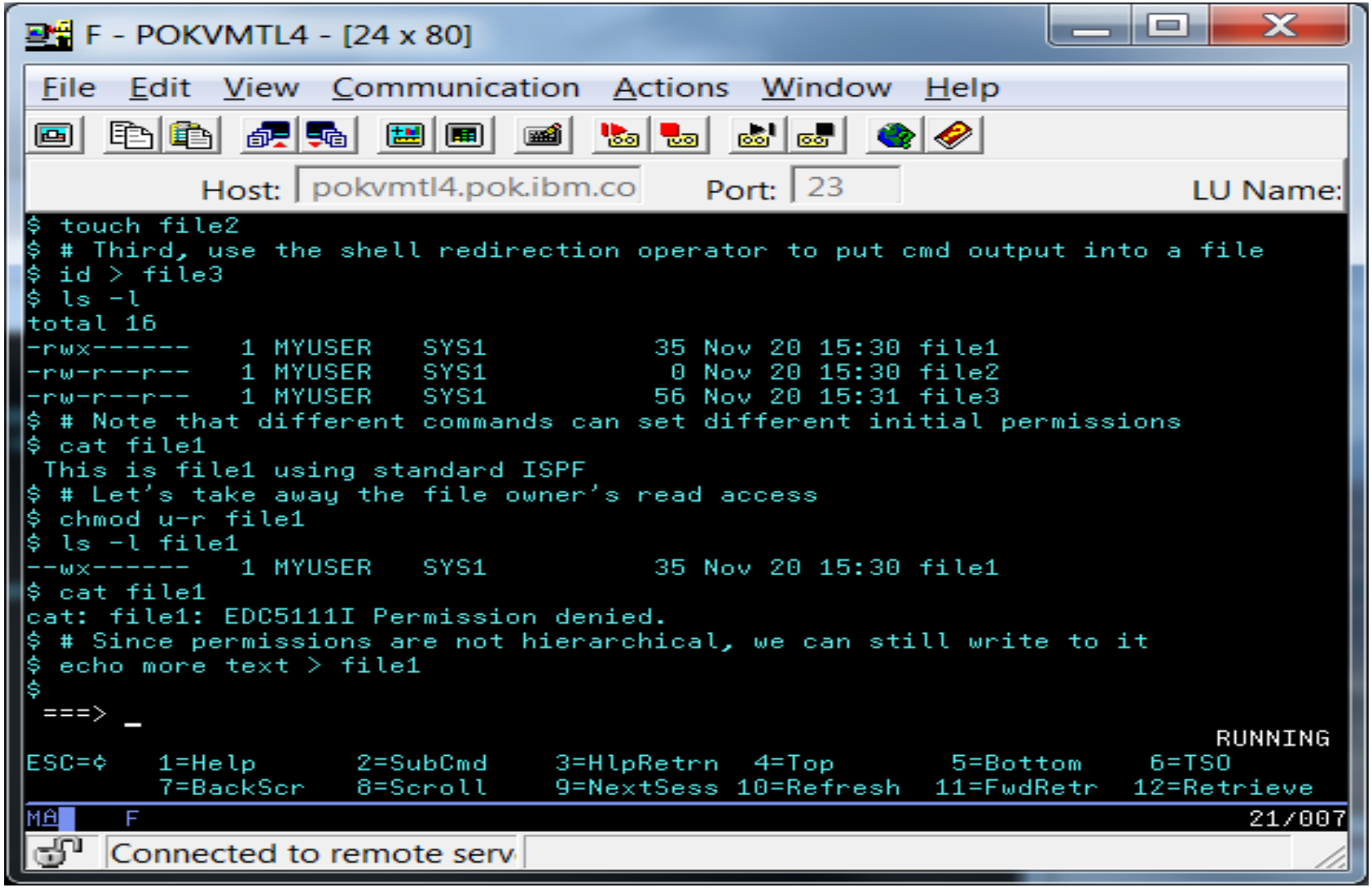
Who controls umask?

- 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

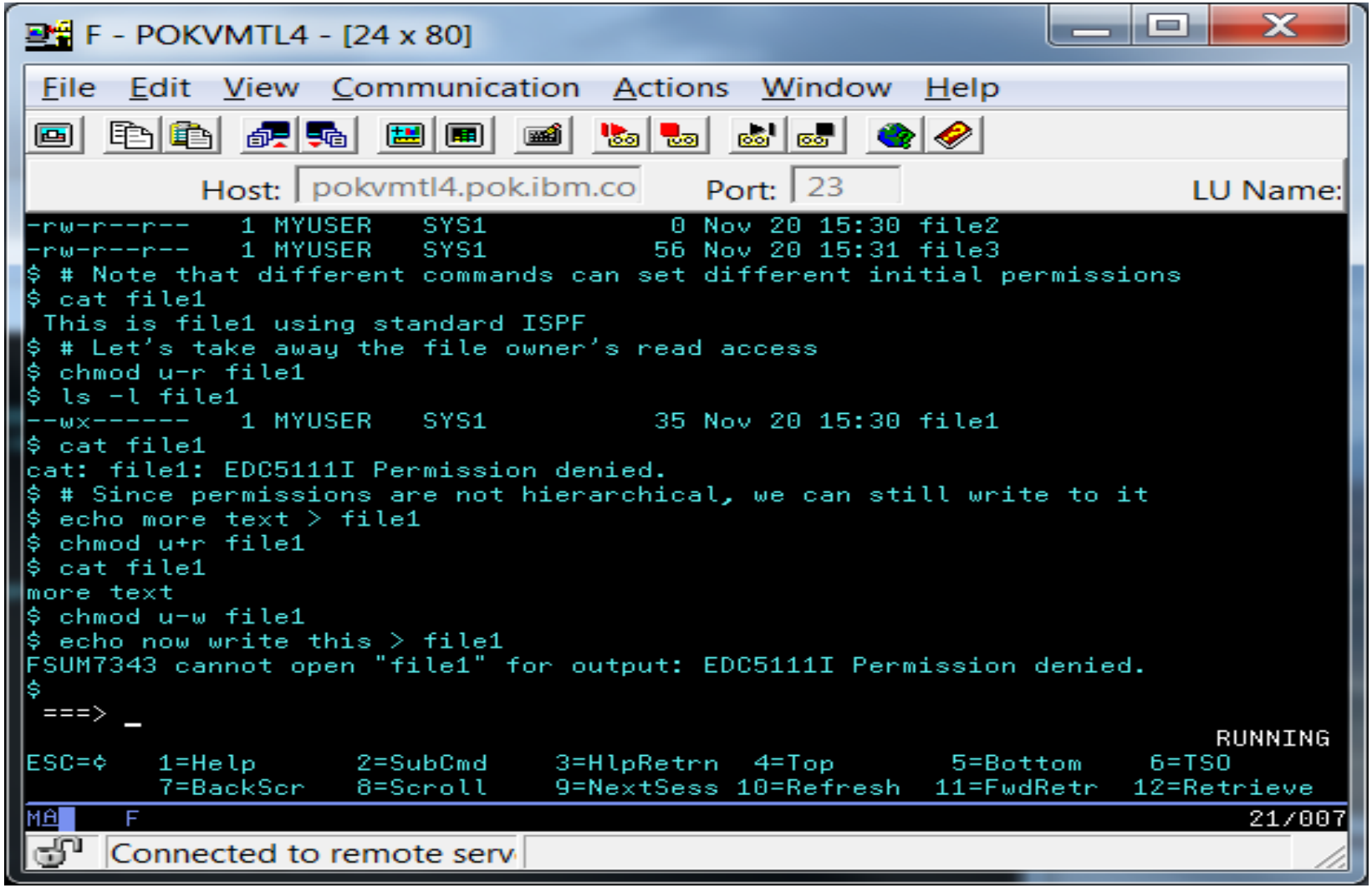
Changing permissions with the chmod command

- 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



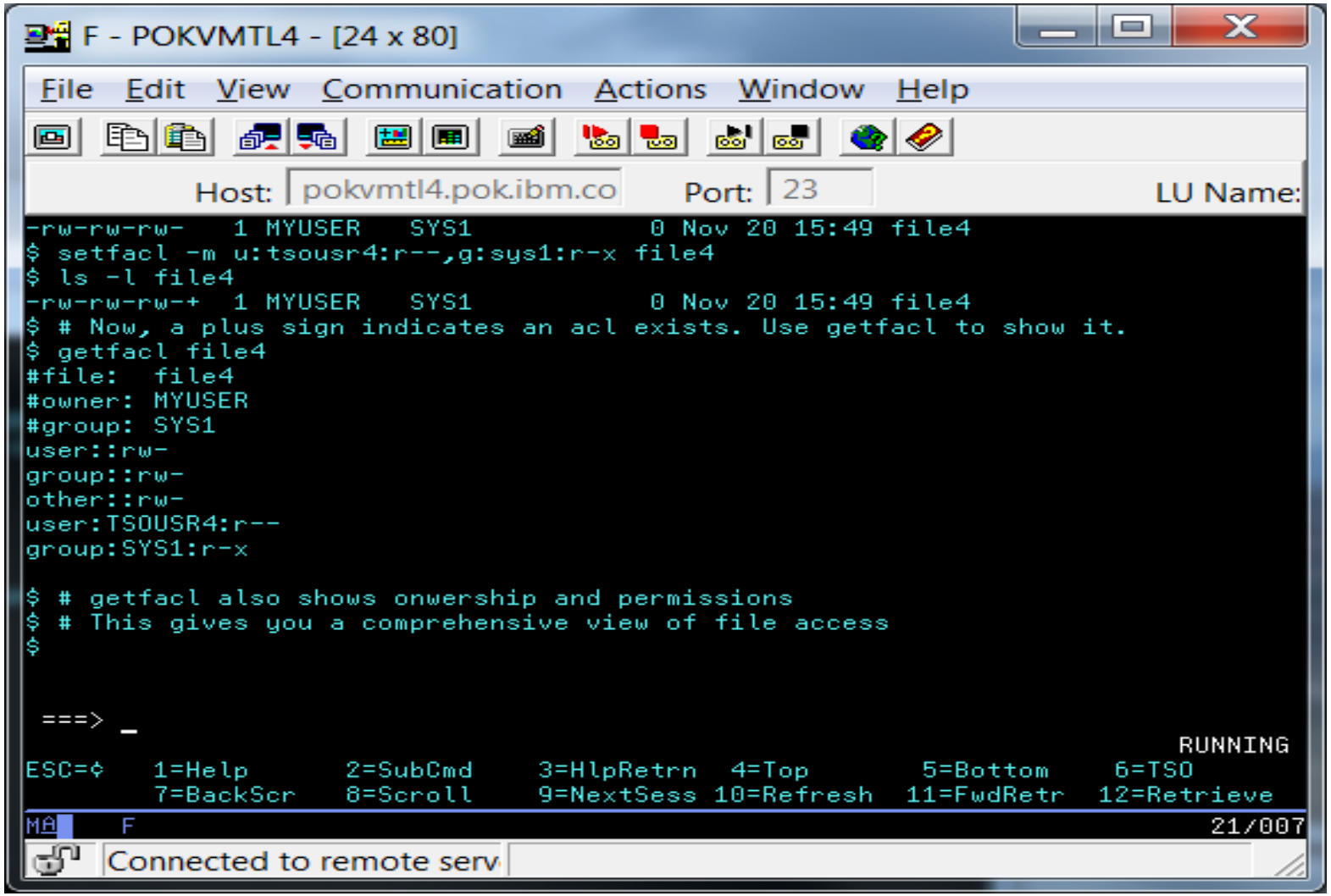
Remove our write access



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



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

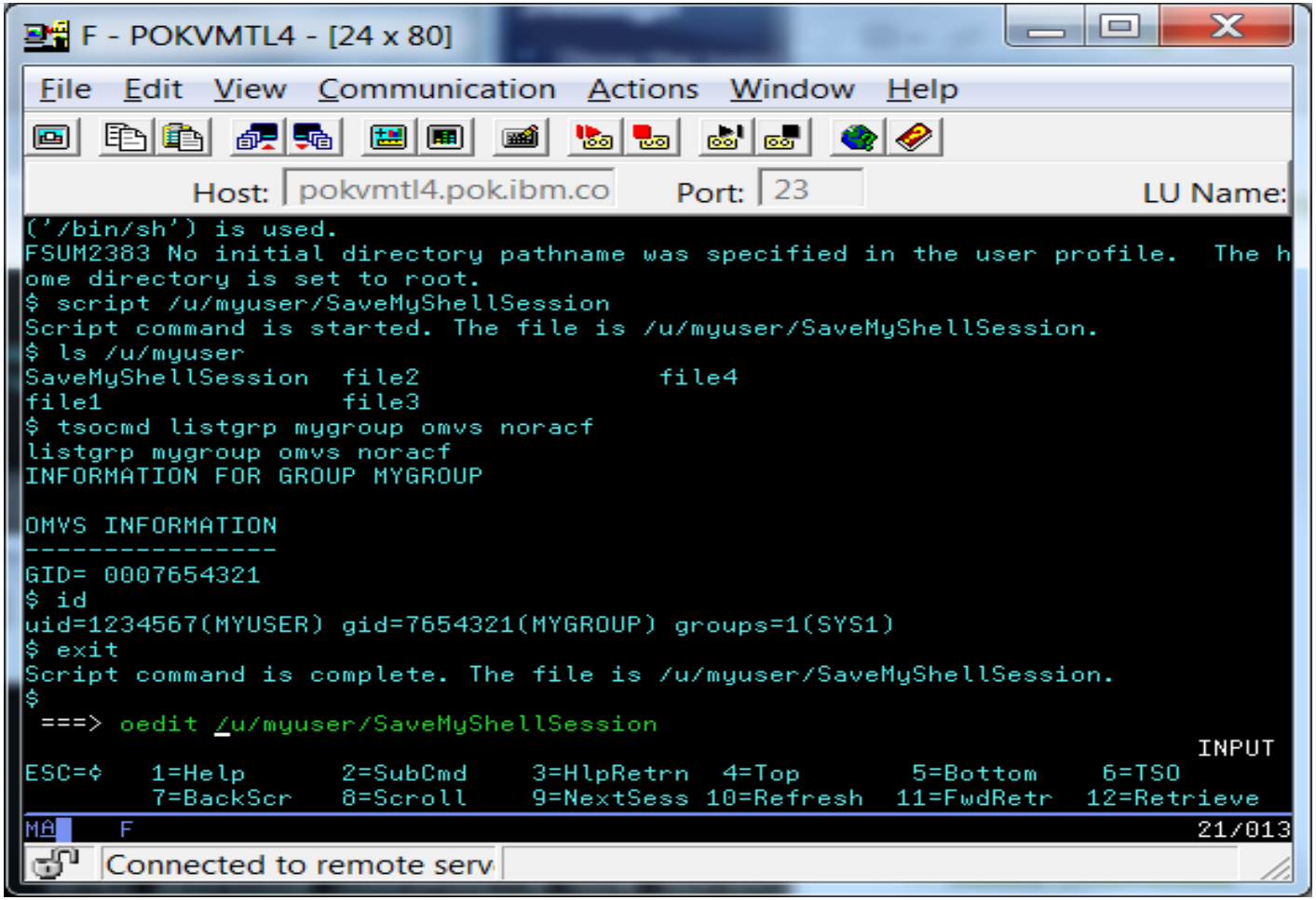
```
-rwx------  --s-  1 MYUSER   SYS1          25 Oct 17 09:36 file1
-rw-r--r--  --s-  1 MYUSER   SYS1           0 Oct 17 09:37 file2
-rw-r--r--  --s-  1 MYUSER   SYS1          56 Oct 17 09:39 file3
```

- 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 fouth 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

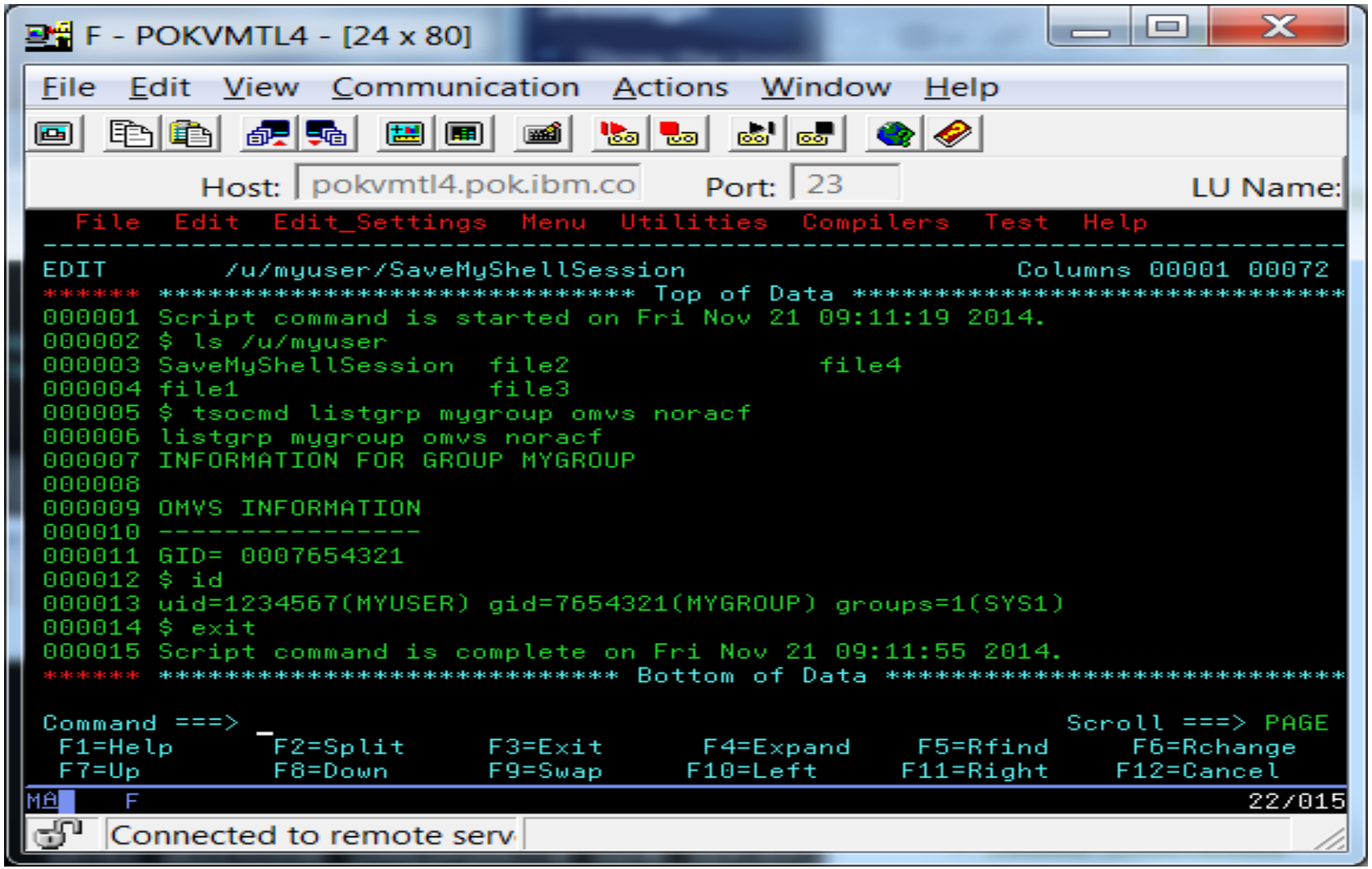
Bonus tip of the day – save your shell session to a file!

- Discovered by yours truly just after the nick of time

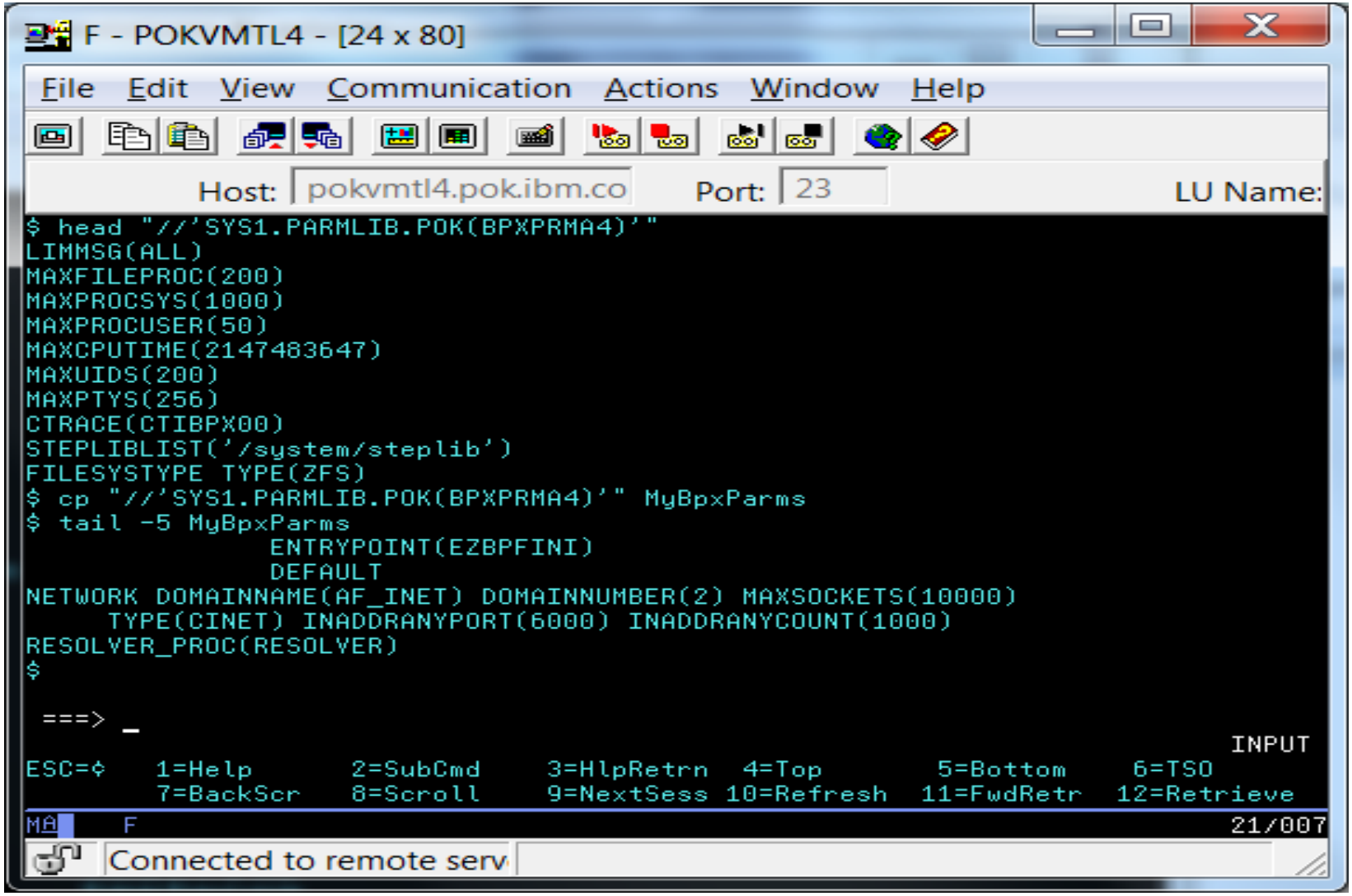


The script command was introduced in z/OS V1R13

The fruits of our labor captured for posterity



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



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