CHAPTER 1

Introduction to the Mainframe

The mainframe is the backbone of many industries that are the lifeblood of the global economy. More mainframe processing power is being shipped now than has ever been shipped. Businesses that require unparalleled security, availability, and reliability for their "bet your business" applications depend on the IBM zSeries® mainframe, which runs the z/OS operating system and is protected by the IBM Resource Access Control Facility (RACF).

In this book, we explain the basics of z/OS, focusing on z/OS security and RACF. This chapter describes the evolution of the mainframe and the reasons it is the leading platform for reliable computing. It also explains how to use the key elements of z/OS.

1.1 Why Use a Mainframe?

This book introduces security administrators to the world of z/OS. We expect that you already have experience with Linux, UNIX, or Windows. Using this prerequisite knowledge, we teach you how to use the mainframe and how to configure RACF, the security subsystem. At the end of each chapter, we list sources for additional information.

If you are the kind of person who wants to go right to typing commands and seeing results, skip on over to Section 1.2, "Getting Started," to learn about the z/OS Time Sharing Option (TSO) environment. However, we recommend that you read the rest of this section to understand the mainframe design philosophy. Many of the differences between the mainframe and other operating systems only make sense if you understand the history and philosophy behind mainframes.

1.1.1 A Little History

Few industries have had the rapid, almost explosive growth that we have seen in the information technology industry. The term *computer* originally referred to people who did manual calculations. The earliest nonhuman computers were mechanical devices that performed mathematical computations. Mechanical devices evolved into vacuum tube devices, which, in turn, were replaced by transistorized computers, which were replaced by integrated circuit devices.

Where do mainframes fit in? The mainframes we use today date back to April 7, 1964, with the announcement of the IBM System/360™. System/360 was a revolutionary step in the development of the computer for many reasons, including these:

- System/360 could do both numerically intensive scientific computing and input/output intensive commercial computing.
- System/360 was a line of upwardly compatible computers that allowed installations to move to more powerful computers without having to rewrite their programs.
- System/360 utilized dedicated computers that managed the input/output operations, which allowed the central processing unit to focus its resources on the application.

These systems were short on memory and did not run nearly as fast as modern computers. For example, some models of the System/360 were run with 32K (yes, K, as in 1,024 bytes) of RAM, which had to accommodate both the application and the operating system. Hardware and software had to be optimized to make the best use of limited resources.

IBM invested $5 billion in the development of the System/360 product line. This was a truly "bet your company" investment. Five billion dollars represented more than one and a half times IBM's total 1964 gross revenue of $3.2 billion. To put it into perspective, given IBM's 2005 gross revenue of $91 billion, an equivalent project would be more than a $140 billion project!

The z/OS operating system that we are discussing here traces itself back to System/360. One of the operating systems that ran on System/360 was OS/360. One variant of OS/360 was MVT (multitasking with a variable number of tasks). When IBM introduced virtual memory with System/370™, the operating system was renamed to SVS (single virtual storage), recognizing that a single virtual address space existed for the operating system and all users. This was quickly replaced with a version of the operating system that provided a separate virtual address space for each user. This version of the operating system was called MVS™ (multiple virtual storage). Later, IBM packaged MVS and many of its key subsystems together (don't worry about what a subsystem is just now…we'll get to that later) and called the result OS/390®, which is the immediate predecessor to z/OS.

1.1.2 Why Are Mainframes Different?

Mainframes were designed initially for high-volume business transactions and, for more than 40 years, have been continually enhanced to meet the challenges of business data processing. No computing platform can handle a diversity of workloads better than a mainframe.

But aren't "insert-your-favorite-alternative-platform" computers cheaper/faster/easier to operate? The answer is: It all depends. A student who is composing his term paper does not have the same information needs as a bank that needs to handle millions of transactions each day, especially because the bank also needs to be able to pass security and accounting audits to verify that each account has the correct balance.

Mainframes aren't for every computing task. Businesses opt for mainframes and mainframe operating systems when they have large volumes of data, large transaction volumes, large data transfer requirements, a need for an extremely reliable system, or many differing types of workloads that would operate best if they were located on the same computer. Mainframes excel in these types of environments.

1.1.3 Mainframe vs. Client/Server

In a client/server architecture, multiple computers typically cooperate to do the same task. For example, in Figure 1.1 the application uses a Web server, a database server, and an LDAP server.

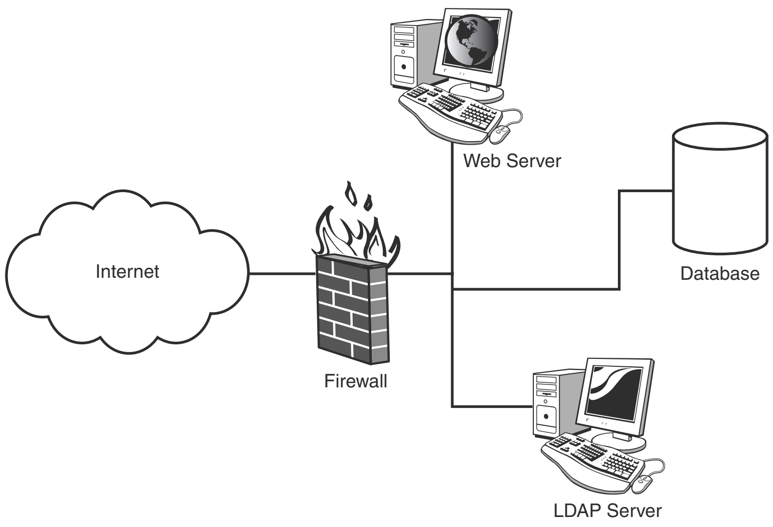


Figure 1.1 Client/server architecture

On a mainframe, the same computer does everything. One security package (RACF, in most cases) protects one operating system kernel. Mainframe subsystems do everything else, as you can see in Figure 1.2.

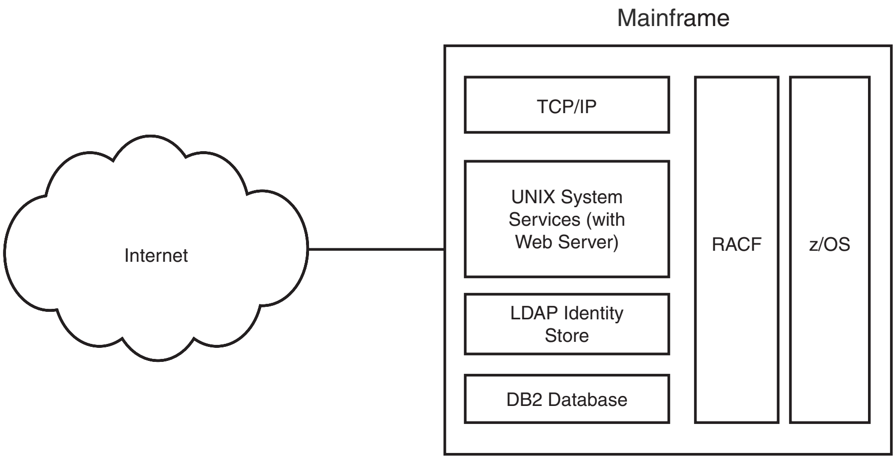


Figure 1.2 Mainframe architecture

That's a little of the "why" of mainframes. Now let's get started with the "how."

1.2 Getting Started

Virtually every computer book starts with a simple example that enables you to get your feet wet. We've got several "Hello, World" examples that will introduce you to:

1. Interactive computing using the z/OS Time Sharing Option (TSO)
2. Batch computing using Job Control Language (JCL)
3. UNIX System Services (USS)

1.2.1 What You Will Need

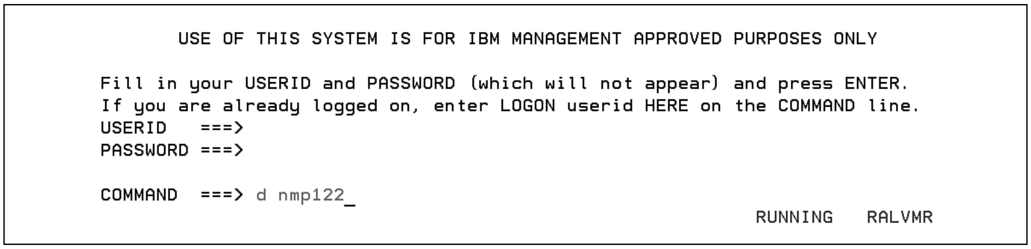
For the purposes of this chapter, you'll need a TSO and OMVS user ID for a z/OS system and the initial password. This user ID is created for you by a system administrator. Your user ID is a one- to seven-character string that is your "handle" for all the work you do within z/OS. It's the basis for your computer identity within z/OS and the anchor point for all your access control permissions.

For the other chapters of this book, you will need your own z/OS image, a copy of the operating system running inside its own virtual machine. On this image, you will need a TSO account with RACF special authority, which corresponds roughly to root under UNIX. Because you will need to change audit settings, it is not enough to have privileges for a specific group within RACF—you need to have global RACF special authority.

1.2.2 Logging in to the Mainframe

In the old days, access to the mainframe was handled mostly by dedicated terminals that were hard-wired to the mainframe. Today, the terminal is a run-of-the-mill PC connected by TCP/IP. The PC runs a program that imitates an old-fashioned terminal.

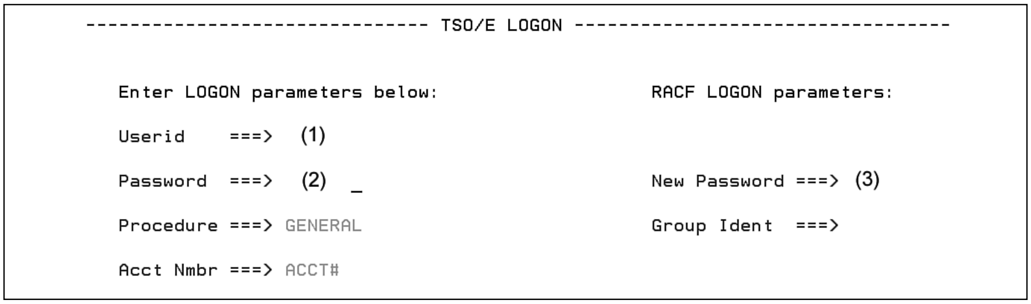
To connect to the mainframe, run the terminal emulator and point it to the IP address of the mainframe and the TCP port number for TSO. After you do that, you might need to "dial" to the correct virtual machine. Figure 1.3 shows a user "dialing" to NMP122, the z/OS 1.6 image used for the screenshots in this book. Some terminal emulators require you to press the right Ctrl key, instead of Enter, to enter a command to the mainframe; this is because the right Ctrl key is located where the Enter key was located on the original 3270 terminal. After you connect to the image, you might need to type **`TSO <your user ID>`** to reach the TSO logon panel.



```
        USE OF THIS SYSTEM IS FOR IBM MANAGEMENT APPROVED PURPOSES ONLY

 Fill in your USERID and PASSWORD (which will not appear) and press ENTER.
 If you are already logged on, enter LOGON userid HERE on the COMMAND line.
 USERID   ===>
 PASSWORD ===>

 COMMAND  ===> d nmp122_
                                                           RUNNING   RALVMR
```

Figure 1.3 The command to dial the correct system

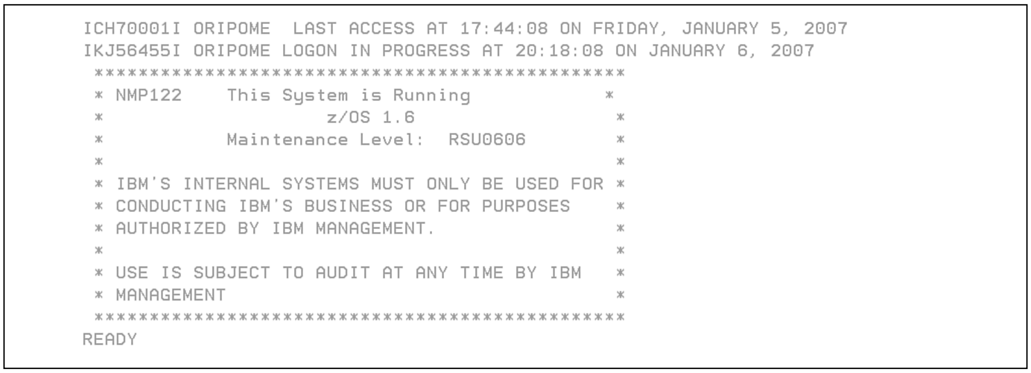
Figure 1.4 shows the TSO logon panel. On this panel, enter the user ID that you've been given in (1) in the figure, your password in (2), and a new password of your choosing in (3). Because the person who created your user ID knows the password, you need to change it to ensure that, from now on, only you can log on to TSO using your user ID. Press Enter to start the logon process.



```
------------------------------- TSO/E LOGON ----------------------------------


  Enter LOGON parameters below:                 RACF LOGON parameters:

  Userid    ===>  (1)

  Password  ===>  (2)  _                        New Password ===>  (3)

  Procedure ===> GENERAL                        Group Ident  ===>

  Acct Nmbr ===> ACCT#
```

Figure 1.4 TSO logon panel

After a few moments, you'll see lines displayed that look similar to Figure 1.5. The first line tells you the last time your user ID was used. This is an elementary intrusion-detection mechanism: If the date and the time do not look correct, you should call your security department to investigate who is using your user ID without your permission.



```
ICH70001I ORIPOME  LAST ACCESS AT 17:44:08 ON FRIDAY, JANUARY 5, 2007
IKJ56455I ORIPOME LOGON IN PROGRESS AT 20:18:08 ON JANUARY 6, 2007
 *************************************************
 * NMP122    This System is Running             *
 *                   z/OS 1.6                    *
 *           Maintenance Level:  RSU0606         *
 *                                               *
 * IBM'S INTERNAL SYSTEMS MUST ONLY BE USED FOR *
 * CONDUCTING IBM'S BUSINESS OR FOR PURPOSES     *
 * AUTHORIZED BY IBM MANAGEMENT.                 *
 *                                               *
 * USE IS SUBJECT TO AUDIT AT ANY TIME BY IBM    *
 * MANAGEMENT                                    *
 *************************************************
READY
```

Figure 1.5 TSO logon results

The second line tells you how long you have until you will need to change your password. A good security policy requires that you change your password periodically. Your installation's policy is enforced whenever you enter the system.

The next line tells you that you have been authenticated (that is, your password is correct and you have not been denied access to the system for any other reason), and now TSO starts to build your logon environment.

This is followed by an installation-specific message, usually reminders of important aspects of your installation's information policy.

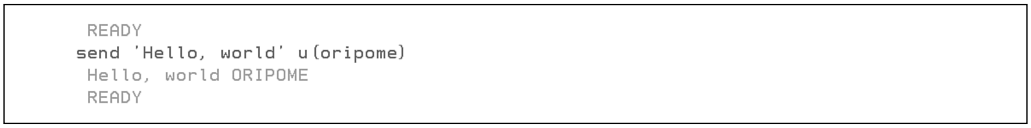
Some installations take users immediately into ISPF, the menu-driven system you will later see in Figure 1.8. In that case, type **`=x`** to exit into TSO so you can run the next exercise.

1.2.3 "Hello, World" from TSO

When this is done, you'll see `READY`. This is the TSO command prompt, similar to `C:\>` under Windows. It's time for our simplistic, trivial, yet traditional, "Hello, World" example. We'll use the `SEND` command to send a message with the text "Hello, World" to a user. Think of `SEND` as TSO's instant messenger (IM). Because the only user that you know right now is yourself, you will be the originator of the message as well as the recipient. Ready (pun intended)? Type this:

```
send 'Hello, World' u(<your user name>)
```

As you can see in Figure 1.6, TSO echoes what you typed. The `SEND` command processor sends the message to the intended recipient, the user ID ORIPOME. After the `SEND` command, TSO prompts you with `READY` to let you know that you can enter more commands.

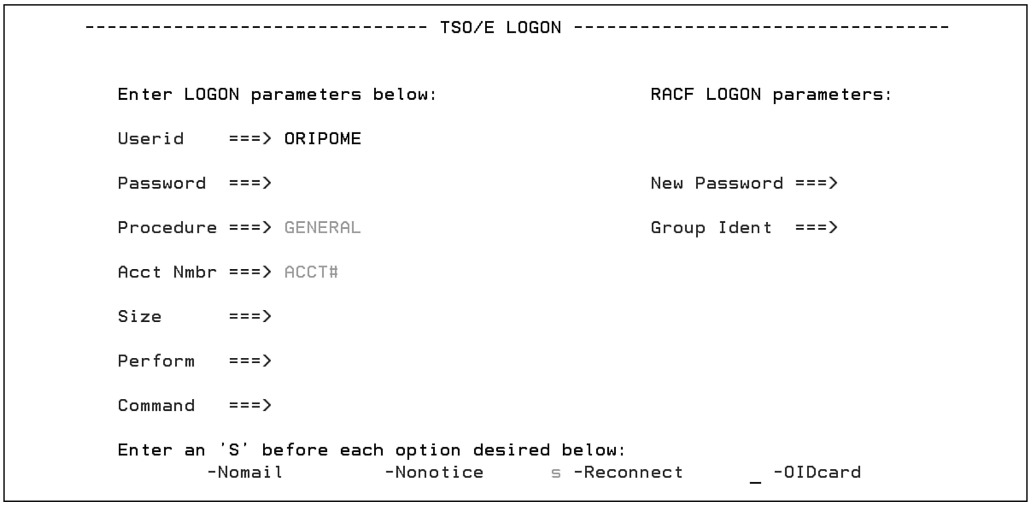


```
 READY
send 'Hello, world' u(oripome)
 Hello, world ORIPOME
 READY
```

Figure 1.6 "Hello, World" from TSO

Congratulations! You've logged on to TSO and said hello to the world. Note that the only person who saw your exclamation was you, so feel free to experiment with other (business-appropriate, of course!) phrases.

When you are done with the mainframe, you need to log off, using the `logoff` command. If you just close the terminal emulator, the session remains open. If you already closed the terminal emulator and you need to log on while you have a running session, type **`s`** before the `Reconnect` option, as shown in Figure 1.7.



```
------------------------------- TSO/E LOGON ----------------------------------


   Enter LOGON parameters below:                 RACF LOGON parameters:

   Userid    ===> ORIPOME

   Password  ===>                                New Password ===>

   Procedure ===> GENERAL                        Group Ident  ===>

   Acct Nmbr ===> ACCT#

   Size      ===>

   Perform   ===>

   Command   ===>

   Enter an 'S' before each option desired below:
           -Nomail         -Nonotice      s -Reconnect       _ -OIDcard
```

Figure 1.7 TSO logon panel with Reconnect

1.3 Job Control Language (JCL)

Entering commands from TSO is one way to accomplish tasks in z/OS, but many other ways exist. One of the most popular and powerful ways is to create files that contain lists of things to do. These lists are called batch jobs and are written in z/OS Job Control Language (JCL), which fulfills roughly the same role as shell scripting languages in UNIX.

1.3.1 Introduction to JCL

JCL is a language with its own unique vocabulary and syntax. Before you can write your first JCL, you need to understand a few z/OS concepts and facilities.

We use JCL to create batch jobs. A batch job is a request that z/OS will execute later. z/OS will choose when to execute the job and how much z/OS resources the job can have based upon the policies that the system administrator has set up. This is a key feature of z/OS: z/OS can manage multiple diverse workloads (jobs) based upon the service level that the installation wants. For example, online financial applications will be given higher priority and, therefore, more z/OS resources, and noncritical work will be given a lower priority and, therefore, fewer z/OS resources. z/OS constantly monitors the resources that are available and how they are consumed, reallocating them to meet the installation goals. We could spend volumes describing just this one feature of z/OS, but this book is supposed to be about security, so we won't.

In your batch job, you will tell z/OS this information:

- You'll give the name of your job, with a `//JOB` statement
- You'll specify the program you want to execute, with a `//EXEC PGM=<program name>` statement
- If your program uses or creates any data, you'll point to the data using a `//DD` statement.

Listing 1.1 shows a trivial JCL job. Don't worry about executing this job, or about the exact meaning of each word—we explain them later in this chapter.

Listing 1.1 Trivial Batch Job

```
//MARKNJ JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
//       EXEC PGM=IEFBR14
```

This job executes an IBM-provided z/OS program called IEFBR14. This is a dummy program that tells z/OS "I'm done and all is well." It requires no input and produces no output other than an indication to the operating system that it completed successfully.

You can also run TSO as a batch job by using JCL to tell z/OS this information:

- The name of the job
- The program to run, which is the TSO interpreter IKJEFT01
- Where to get the input for IKJEFT01 and the commands that you want to execute
- Where to put the output from IKJEFT01, the output from TSO, and the commands that you execute

Listing 1.2 shows a batch job that runs TSO to send a message.

Listing 1.2 Batch Job That Sends a Message Using TSO

```
//TSOJOB   JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
//         EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
SEND 'Hello, World' U(ORIPOME)
/*
```

1.3.2 Data Sets

To submit a batch job, you need to understand data sets. As the name implies, a data set is a set or collection of data. Data sets are made up of records. To improve performance, records can be gathered together into blocks. Data sets fill the same function as files and directories in UNIX and Windows.

When you create a data set, you assign it a name. The name can be up to 44 characters long and consists of multiple parts, separated by dots (`.`). Each part can be up to eight characters. In a RACF-protected system, the first qualifier is either a user ID or a group name. We discuss group names in Chapter 2, “Users and Groups.”

Note

This means that in a z/OS system protected by RACF, each data set belongs to either a user or a group. This is different from the situation in UNIX and Linux, where each file has a user and a group. We explain the meaning of data set ownership in Chapter 3, “Protecting Data Sets and Other Resources.”

Examples of valid data set names are

- `MARKN.BOOK.CNTL`
- `ORI.LONG$$$$.DATASET.NAME.WITHLOTS.OFQUALS`
- `SYS1.PARMLIB`

Examples of data set names that are invalid are

- `MARKN.QUALIFIERTOOLONG.CNTL` (the middle qualifier is longer than eight characters)
- `ORI.THIS.DATA.SET.NAME.IS.WAY.WAY.WAY.TOO.LONG` (the total data set name is longer than 44 characters)

The act of creating a data set is called data set allocation. To allocate a data set, you need to tell z/OS a few things about the data set:

- The length of records within the data set expressed in bytes (often called the LRECL)
- The expected size of the data set
- If records are to be blocked, the number of bytes in the block (called the BLKSIZE)
- The organization of the data set (referred to as the DSORG)

Data set organization requires a little explanation. z/OS allows you to define a data set that is partitioned into multiple "mini data sets" called members. This type of data set is called a partitioned data set (PDS). PDSs contain a directory that tells z/OS the name of the member as well as how to locate the member, similar to directories under UNIX, Windows, and Linux. Much of the work that you do in z/OS involves the use of PDS data sets, or their more modern version, the extended PDS called the PDSE or library.

In contrast to UNIX, Linux, and Windows, z/OS requires you to specify the maximum size of each data set, for two reasons. The first is historical—z/OS is backward compatible and can run applications that were developed 40 years ago when disk space was at a premium. The second reason is that z/OS is designed for high-availability applications. When you specify the maximum size of each data set, you can ensure that the important data sets will always have the disk space they need. For simple data sets, such as the ones that we are discussing here, the allocation consists of two parts:

1. The initial size of the data set is called the primary extent. This is the amount of space that z/OS reserves for the data set right now. If you think that your data set might grow in size later, you can specify the size of the secondary extents.
2. If the data set is expected to grow beyond its initial size, additional allocations of disk storage can be given to the data set by specifying the size of the secondary extent. If the primary extent of your data set fills up, z/OS allocates the secondary extent up to 15 times. This allows your data set to grow gradually up to the maximum data set size.

When defining the size of the primary and secondary extents, you can do it in bytes or based on the device geometry in units of space called tracks or cylinders. Understanding these two terms requires understanding how a disk drive works. A disk drive consists of a set of rotating metallic platters upon which data is stored magnetically. Data is written on the disk in sets of concentric circles. Each of these circles is called a track. If you project that track from the top of the stack of platters to the bottom, you have created a cylinder. It is faster to read information that is stored in the same cylinder than information that is spread across cylinders.

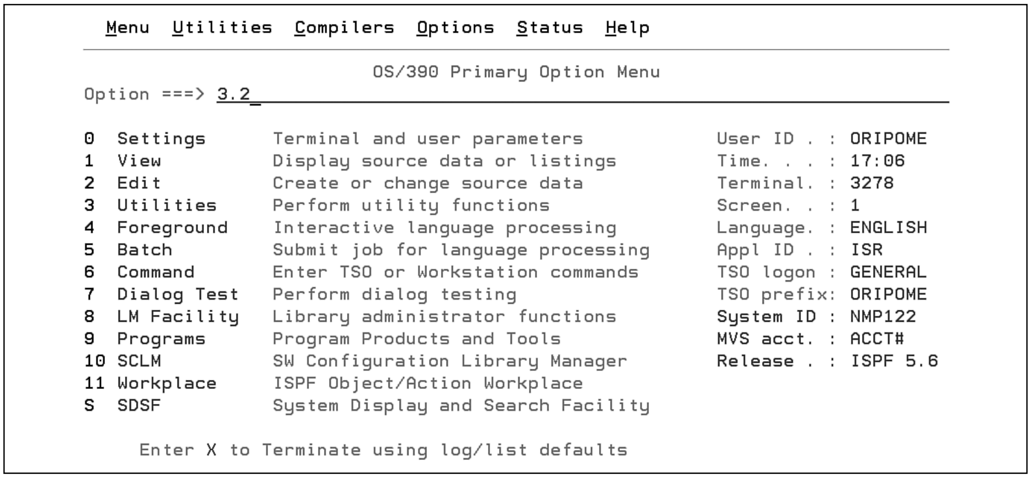
1.3.3 Using ISPF to Create and Run Batch Jobs

Before we can create and submit a batch job, we need to create a data set to hold it. The simplest way to allocate a data set is to use the Interactive System Productivity Facility (ISPF).

1.3.3.1 Data Set Creation

Getting into ISPF is very simple: just type ISPF on the TSO command line. ISPF enables you to perform many common z/OS tasks from a full-screen interactive dialog. You move about the ISPF dialogs by specifying the number of the dialog that you want to use. For example, Utilities is option 3. You can then choose the suboption, which enables you to define and delete data sets. That's option 2. We often combine these two and type them as ISPF option 3.2.

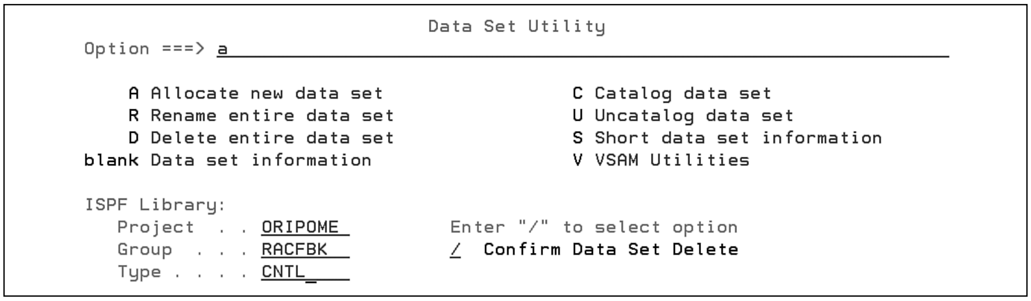
As you can see in Figure 1.8, each ISPF panel presents the list of options that you can select. When you get familiar with ISPF, you can use ISPF's fast-path feature and specify **`=3.2`** from any ISPF panel to have ISPF take you directly to the data set allocate and delete panel.



```
  Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                          OS/390 Primary Option Menu
 Option ===> 3.2_

 0  Settings      Terminal and user parameters          User ID . : ORIPOME
 1  View          Display source data or listings       Time. . . : 17:06
 2  Edit          Create or change source data          Terminal. : 3278
 3  Utilities     Perform utility functions             Screen. . : 1
 4  Foreground    Interactive language processing       Language. : ENGLISH
 5  Batch         Submit job for language processing   Appl ID . : ISR
 6  Command       Enter TSO or Workstation commands     TSO logon : GENERAL
 7  Dialog Test   Perform dialog testing                TSO prefix: ORIPOME
 8  LM Facility   Library administrator functions       System ID : NMP122
 9  Programs      Program Products and Tools            MVS acct. : ACCT#
 10 SCLM          SW Configuration Library Manager      Release . : ISPF 5.6
 11 Workplace     ISPF Object/Action Workplace
 S  SDSF          System Display and Search Facility

      Enter X to Terminate using log/list defaults
```

Figure 1.8 Main menu of ISPF

Select option **3.2** and press Enter (or the right Ctrl key). ISPF now takes you to a panel where you can allocate and delete data sets. Type **`A`** as the option, your user ID as the project (`ORIPOME` in the screenshot), **`RACFBK`** as the group, and **`CNTL`** as the type, as shown in Figure 1.9. By convention, CNTL is used for data sets that store JCL jobs, which correspond roughly to shell scripts or batch files.



```
                              Data Set Utility
 Option ===> a

     A Allocate new data set                 C Catalog data set
     R Rename entire data set                U Uncatalog data set
     D Delete entire data set                S Short data set information
 blank Data set information                  V VSAM Utilities

 ISPF Library:
    Project  . . ORIPOME          Enter "/" to select option
    Group  . . . RACFBK           /  Confirm Data Set Delete
    Type . . . . CNTL_
```

Figure 1.9 Step 1 in data set creation

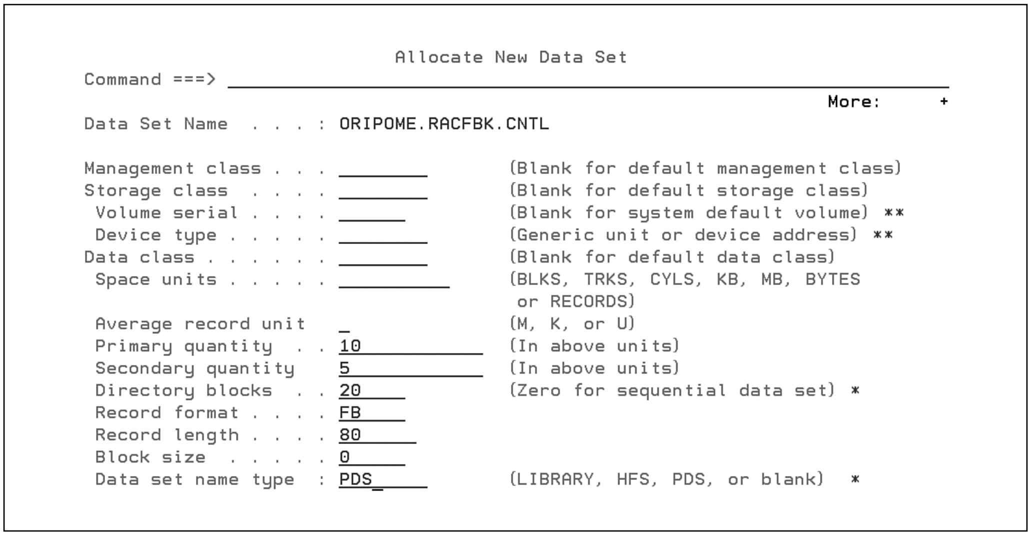
To allocate the data set, you need to tell z/OS this information:

- The expected size of your data set. We'll be adding other members to this partitioned data set, so let's give it an initial size (primary allocation) of ten tracks and allow it to grow five tracks at a time (secondary allocation). Remember that z/OS uses the secondary allocation 15 times before the data set reaches its maximum size.
- The length of each record in your data set. One of the most common record lengths in z/OS is 80 bytes, which is what we will use for our first few data sets.
- The size of each block. For performance reasons, you might want to tell z/OS that whenever it reads a record, it should read a group of them. That way, when you read the next record, it will already be in memory. Every time z/OS reads from the disk, it reads an entire block. The block size that you select also affects the efficiency of the records stored on the disk drive. If you specify `0`, z/OS calculates the best block size for the device upon which the data set is placed.
- The number of directory blocks. When a data set is a partitioned data set, you need to tell z/OS how much space on the data set should be reserved for the directory. Each directory block has enough space to hold the information for about five members. We'll specify 20 blocks, which will give us plenty of space for new members.
- The organization of the data set. Many different types of data sets exist. For our purposes, we'll be working with two types of data sets: normal data sets (called sequential data sets) and partitioned data sets. For this data set, specify PDS for a partitioned data set. Sequential data sets are similar to files under other operating systems. Partitioned data sets contain multiple members, distinguished by name. Each member is similar to a file, so the entire partitioned data set is similar to a directory.

After you have typed all this information, your panel should look similar to Figure 1.10. Press Enter to create the data set. ISPF responds by representing the Data set utility panel with `Data Set Allocated` highlighted in the upper-right corner.

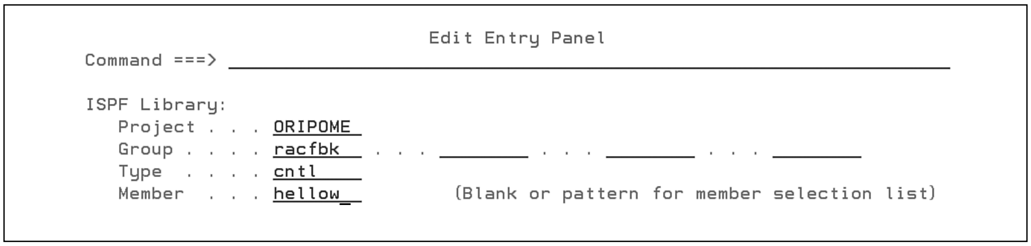
1.3.3.2 Editing Data Set Members

When the data set is created , go to the ISPF editor. To do this, enter **`=2`** on any command line. This is the ISPF "fast path" to the ISPF edit panel, which is option 2 from the main ISPF menu. On this panel, specify the name of the data set that you just allocated. Because you are editing a PDS, you need to specify either the name of an existing member or the name of a member that you want created, as shown in Figure 1.11. In this example, we're creating a member named HELLOW.



```
                          Allocate New Data Set
Command ===> _________________________________________________________________
                                                                  More:     +
Data Set Name  . . . : ORIPOME.RACFBK.CNTL

Management class . . . ________       (Blank for default management class)
Storage class  . . . . ________       (Blank for default storage class)
 Volume serial . . . . ______         (Blank for system default volume) **
 Device type . . . . . ________       (Generic unit or device address) **
Data class . . . . . . ________       (Blank for default data class)
 Space units . . . . . __________     (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit   _              (M, K, or U)
 Primary quantity  . . 10__________   (In above units)
 Secondary quantity    5___________   (In above units)
 Directory blocks  . . 20____         (Zero for sequential data set) *
 Record format . . . . FB____
 Record length . . . . 80_____
 Block size  . . . . . 0_____
 Data set name type  : PDS_____       (LIBRARY, HFS, PDS, or blank)  *
```

Figure 1.10 Step 2 in data set creation



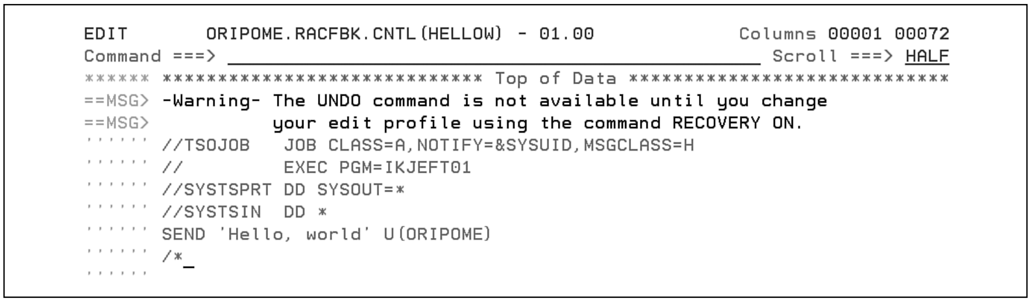
```
                               Edit Entry Panel
Command ===> _________________________________________________________________

ISPF Library:
   Project . . . ORIPOME_
   Group . . . . racfbk__  . . . ________  . . . ________  . . . ________
   Type  . . . . cntl____
   Member  . . . hellow__        (Blank or pattern for member selection list)
```

Figure 1.11 ISPF edit panel

The ISPF Editor

A full description of the ISPF editor with all its features is beyond the scope of this book. For a detailed explanation of the commands, browse to http://publibz.boulder.ibm.com/bookmgr_OS390/libraryserver/zosv1r6/ and open z/OS V1R6.0 Elements, Features, and Software Products → z/OS Elements and Features, z/OS Collection Kit, March 2005 → z/OS V1R6.0 ISPF Edit and Edit Macros → 1.0 Part 1, The ISPF Editor.

After you press Enter, ISPF creates the member and places you in the ISPF editor. At this point, type the JCL shown in Listing 1.2. You need to type it on the lines that start with `''''''`, under the blue asterisks (`*`), as shown in Figure 1.12. Remember to change `ORIPOME` to your own user ID. Traditionally, JCL lines use the eight characters after the `//` for identifiers or leave them empty when no identifier is required. That is the reason, for example, that the word `EXEC` on the second line starts on the same column as the word `JOB` on the first line. The JCL would work with just one space, but it is more readable this way.

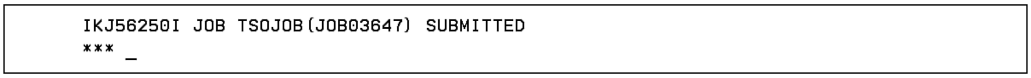


```
EDIT       ORIPOME.RACFBK.CNTL(HELLOW) - 01.00            Columns 00001 00072
Command ===> ________________________________________________ Scroll ===> HALF
****** ***************************** Top of Data ****************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' //TSOJOB   JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
'''''' //         EXEC PGM=IKJEFT01
'''''' //SYSTSPRT DD SYSOUT=*
'''''' //SYSTSIN  DD *
'''''' SEND 'Hello, world' U(ORIPOME)
'''''' /*_
''''''
```

Figure 1.12 The editor after typing the batch job

After this is done, press Enter. ISPF saves your changes and replaces the quotes on the left with line numbers.

At this point, you're ready to submit your job. Type **`SUBMIT`** on the command line, and your batch job is submitted to the job entry subsystem at your installation. You will get a confirmation message with the job number, as shown in Figure 1.13.

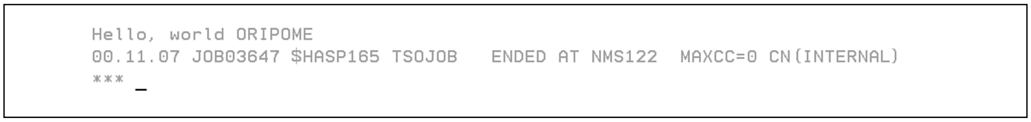


```
IKJ56250I JOB TSOJOB(JOB03647) SUBMITTED
*** _
```

Figure 1.13 Job submission confirmation message

Your installation has a policy for executing batch jobs, and that policy determines when your batch job is executed. After it has executed, you can view the output of the job. When your job executes, it sends a message to your TSO user ID. If you are logged on and are accepting messages, the message appears as your batch job executes. If you are not logged on or are not accepting messages, it is saved and displayed when you next log on.

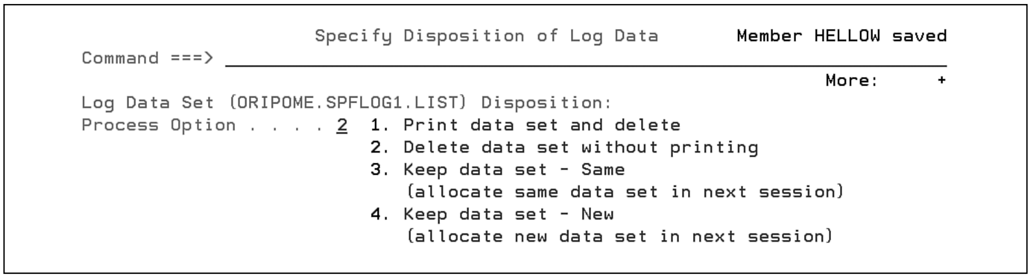
When you see the confirmation message, press Enter again. In all likelihood, your job will have already executed and you will see the message, as well as a job confirmation message, as shown in Figure 1.14.



```
Hello, world ORIPOME
00.11.07 JOB03647 $HASP165 TSOJOB   ENDED AT NMS122  MAXCC=0 CN(INTERNAL)
*** _
```

Figure 1.14 The message the job sent

When you are done with ISPF, enter **=x** on the command line to tell it to exit. If you get a log data panel, such as the one in Figure 1.15, select option **2** to delete the log. You can then use **LOGOFF** to exit TSO.



```
                     Specify Disposition of Log Data       Member HELLOW saved
Command ===> ________________________________________________________________
                                                                More:     +
Log Data Set (ORIPOME.SPFLOG1.LIST) Disposition:
Process Option . . . . 2  1. Print data set and delete
                          2. Delete data set without printing
                          3. Keep data set - Same
                             (allocate same data set in next session)
                          4. Keep data set - New
                             (allocate new data set in next session)
```

Figure 1.15 The log data panel when leaving ISPF

1.3.4 JCL Syntax

Now that you've run the JCL and seen that it works, let's review Listing 1.2 line by line and explain exactly what it does.

First, you'll notice that most lines start with two slashes. The two slashes mark a line as part of JCL. Lines that do not contain those slashes, such as the last two lines in this job, are usually embedded input files.

```
//TSOJOB  JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
```

This line is the job header. It defines a job called `TSOJOB`. The `CLASS` parameter specifies the job's priority, the maximum amount of resources the job is allowed to consume, and so on. `A` is a good default in most installations, at least for the short jobs we'll use in this book.

The `NOTIFY` parameter specifies that a user should be notified when the job ends. It could be the name of a user to notify, but here it is `&SYSUID`, which is a macro that expands to the user who submits the job.

The `MSGCLASS` parameter specifies that the output of the job needs to be held. This makes it accessible afterward, as you will see in Section 1.3.5, "Viewing the Job Output."

```
//        EXEC PGM=IKJEFT01
```

This line starts an execution step—a step in the batch job that runs a program. It is possible for these steps to be named using an identifier immediately after the two slashes. However, this is a very simple job, so there is no need to identify this stage.

The program that this step executes is IKJEFT01, which is the TSO interpreter.

```
//SYSTSPRT DD SYSOUT=*
```

This line is a data definition. It defines the data stream called `SYSTSPRT`, which is the output of TSO. `SYSOUT=*` means that this data stream will go to the standard output of the job. In the next section, you will learn how to get to this output to read it.

```
//SYSTSIN  DD *
```

This line is another data definition. It defines SYSTSIN, which is the input to the TSO interpreter. The value `*` means that the text that follows is the data to be placed in SYSTSIN.

```
SEND 'Hello, World' U(ORIPOME)
/*
```

This is the input to the TSO interpreter. The first line is a command, the same "Hello, World" command we executed in Section 1.2.3, " 'Hello, World' from TSO." The second line, `/*`, is a delimiter that means the end of the file.

1.3.5 Viewing the Job Output

One of the outputs from your batch job was the "Hello, World" that was sent to your TSO ID. Your batch job produced other output as well. What happened to that output? It waits in the system until you or your system administrators tell the system what to do with it.

When you submitted the batch job, it was handed over to the job entry subsystem (JES). JES is responsible for scheduling the job, allocating some of its resources, and managing the job's input and output.

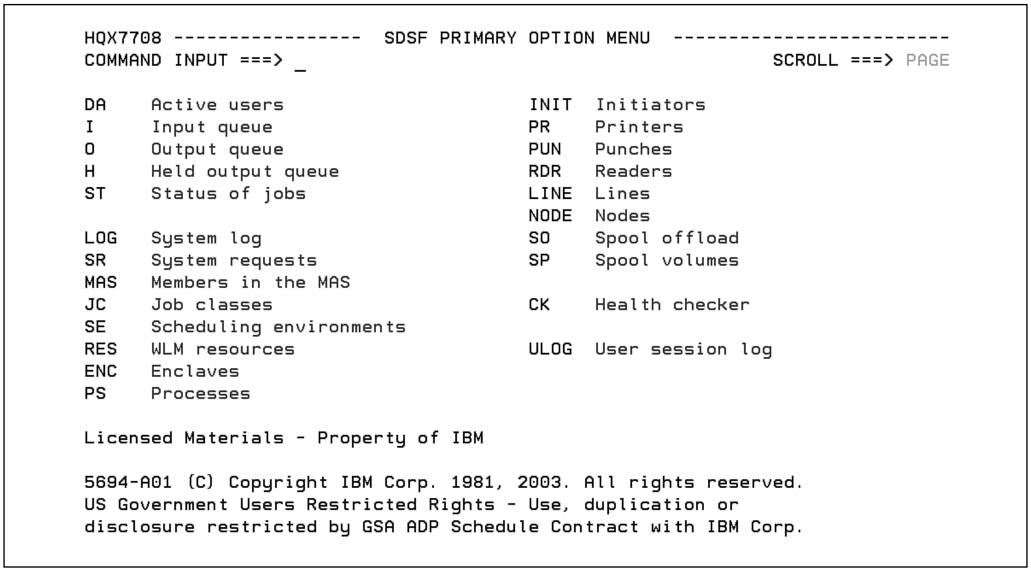
IBM provides job entry subsystems: JES2 and JES3. Most of the z/OS environments use JES2, so our examples are oriented toward it. For those of you who are using JES3, equivalent services exist there.

JES2 and JES3

z/OS has two Job Entry Subsystems (JES): JES2 (part of the base z/OS) and JES3 (an optional add-on). Both provide similar capabilities to manage batch jobs and SYSOUT, the job's output data stream. The primary differences are how systems in a SYSPLEX, a mainframe cluster, are managed. In JES2, each system is a peer, selecting work it can process. In JES3, a control system (the global) passes work to other systems (locals) for processing. JES3 also has additional services to provide additional controls over the timing of job execution. Because each JES has its own set of commands and JCL extensions, it is difficult for an installation to change from one JES to another. As a result, mainframe installations generally run the same JES they have used historically.

One of the most popular ways to view the output of your job is to use the IBM System Display and Search Facility (SDSF) program product. You start up SDSF either as a TSO command (SDSF) or as a dialog from within ISPF. In most installations, SDSF is option `S` from the ISPF Primary Options menu.

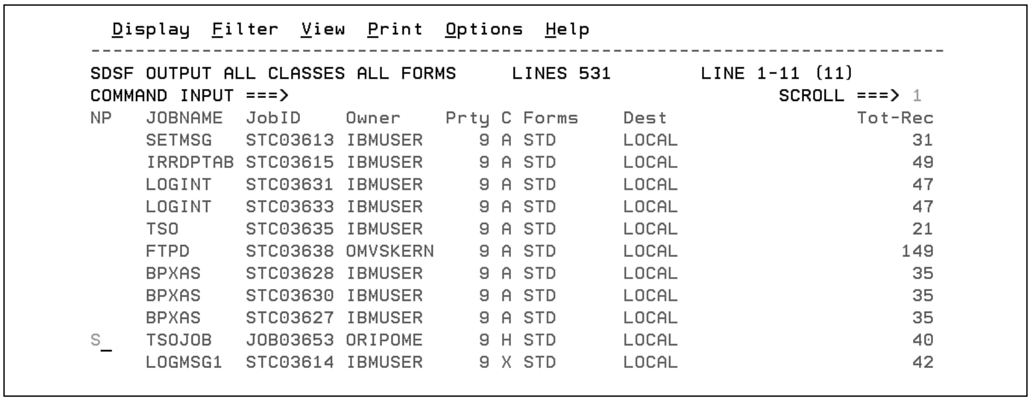
From the ISPF Primary Options menu, select the SDSF option, which brings you to the SDSF Primary Option menu, shown in Figure 1.16. On this panel, the options that are presented depend upon your level of authorization: The more things you are authorized to do, the more options you'll see presented by SDSF on the panel.



```
HQX7708 -----------------  SDSF PRIMARY OPTION MENU  ------------------------
COMMAND INPUT ===> _                                         SCROLL ===> PAGE

DA    Active users                      INIT  Initiators
I     Input queue                       PR    Printers
O     Output queue                      PUN   Punches
H     Held output queue                 RDR   Readers
ST    Status of jobs                    LINE  Lines
                                        NODE  Nodes
LOG   System log                        SO    Spool offload
SR    System requests                   SP    Spool volumes
MAS   Members in the MAS
JC    Job classes                       CK    Health checker
SE    Scheduling environments
RES   WLM resources                     ULOG  User session log
ENC   Enclaves
PS    Processes

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1981, 2003. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

Figure 1.16 The SDSF Primary Option menu

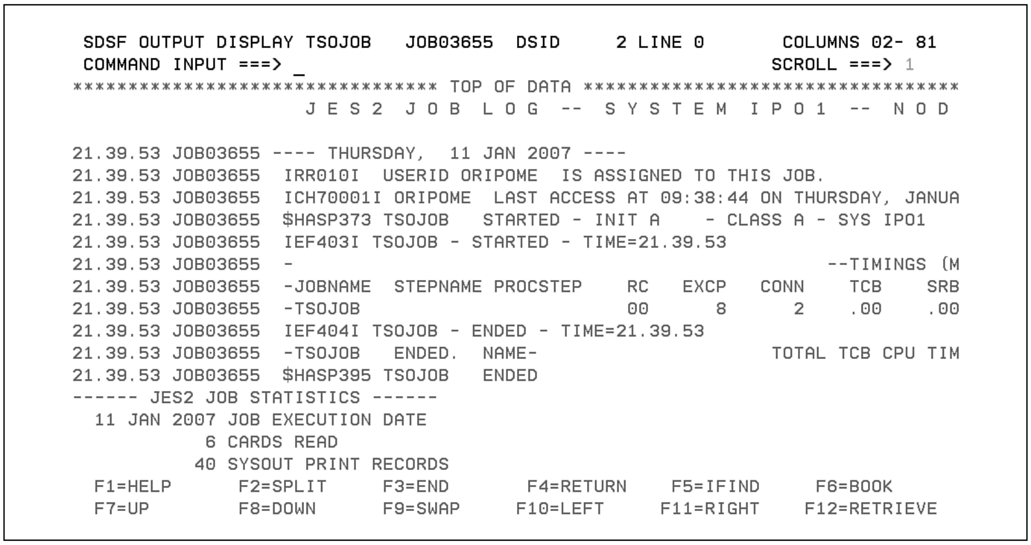
The job's output is in the output queue. Type **O** to enter the output queue, find your job, and type **S** next to it to open the output, as shown in Figure 1.17. If necessary, you can scroll down using F8 or up again using F7.



```
  Display  Filter  View  Print  Options  Help
 -----------------------------------------------------------------------------
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 531        LINE 1-11 (11)
COMMAND INPUT ===>                                            SCROLL ===> 1
NP   JOBNAME  JobID    Owner    Prty C Forms    Dest                 Tot-Rec
     SETMSG   STC03613 IBMUSER     9 A STD      LOCAL                     31
     IRRDPTAB STC03615 IBMUSER     9 A STD      LOCAL                     49
     LOGINT   STC03631 IBMUSER     9 A STD      LOCAL                     47
     LOGINT   STC03633 IBMUSER     9 A STD      LOCAL                     47
     TSO      STC03635 IBMUSER     9 A STD      LOCAL                     21
     FTPD     STC03638 OMVSKERN    9 A STD      LOCAL                    149
     BPXAS    STC03628 IBMUSER     9 A STD      LOCAL                     35
     BPXAS    STC03630 IBMUSER     9 A STD      LOCAL                     35
     BPXAS    STC03627 IBMUSER     9 A STD      LOCAL                     35
S_   TSOJOB   JOB03653 ORIPOME     9 H STD      LOCAL                     40
     LOGMSG1  STC03614 IBMUSER     9 X STD      LOCAL                     42
```

Figure 1.17 The job's output in the output queue

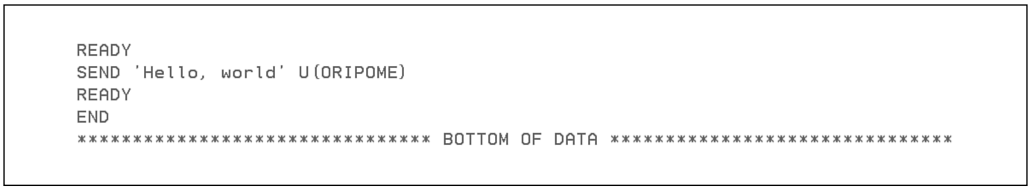
The top part of the output, shown in Figure 1.18, tells when the job started, when it ended, which user ID was assigned to the job, and other job statistics. JES also displays the JCL. Scroll down a page to see more system-generated messages telling you about the resources allocated for your job. You can scroll up (F7), down (F8), left (F10), and right (F11).



```
 SDSF OUTPUT DISPLAY TSOJOB   JOB03655  DSID     2 LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===> _                                          SCROLL ===> 1
********************************* TOP OF DATA *********************************
                    J E S 2  J O B  L O G  --  S Y S T E M  I P O 1  --  N O D

21.39.53 JOB03655 ---- THURSDAY,  11 JAN 2007 ----
21.39.53 JOB03655  IRR010I  USERID ORIPOME  IS ASSIGNED TO THIS JOB.
21.39.53 JOB03655  ICH70001I ORIPOME  LAST ACCESS AT 09:38:44 ON THURSDAY, JANUA
21.39.53 JOB03655  $HASP373 TSOJOB   STARTED - INIT A    - CLASS A - SYS IPO1
21.39.53 JOB03655  IEF403I TSOJOB - STARTED - TIME=21.39.53
21.39.53 JOB03655  -                                                 --TIMINGS (M
21.39.53 JOB03655  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB
21.39.53 JOB03655  -TSOJOB                        00      8      2    .00    .00
21.39.53 JOB03655  IEF404I TSOJOB - ENDED - TIME=21.39.53
21.39.53 JOB03655  -TSOJOB   ENDED.  NAME-                     TOTAL TCB CPU TIM
21.39.53 JOB03655  $HASP395 TSOJOB   ENDED
------ JES2 JOB STATISTICS ------
  11 JAN 2007 JOB EXECUTION DATE
            6 CARDS READ
           40 SYSOUT PRINT RECORDS
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 1.18 The first part of the job's output

The real output of the job is in the last four lines of the job, shown in Figure 1.19. These lines show where we see the batch version of TSO displaying the `READY` prompt, the echoing of the "Hello, World" command, TSO's `READY` response, and the generated `END` statement.



```
READY
SEND 'Hello, world' U(ORIPOME)
READY
END
******************************** BOTTOM OF DATA ******************************
```

Figure 1.19 The output of the job's TSO interpreter

1.3.5.1 Filtering Jobs

A large z/OS installation can have many jobs running at the same time. It is possible to use filtering to see only the jobs that are relevant to you.

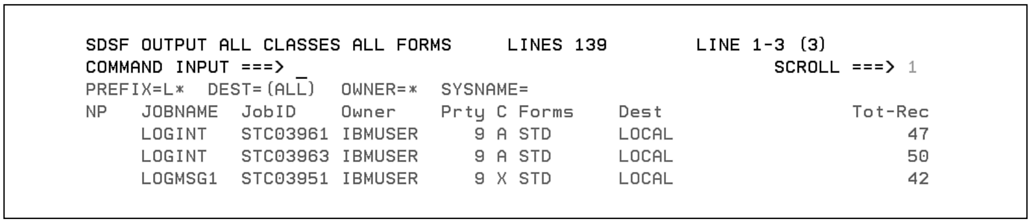
To see the current filters, run this command inside SDSF:

```
SET DISPLAY ON
```

To filter, enter the name of the field to filter (prefix in the job name, destination, owner, or sysname) and the value. For example, this command filters for jobs that start with *L.*

```
PREFIX L*
```

After this command, SDSF will show only those jobs that start with *L,* as you can see in Figure 1.20.



```
SDSF OUTPUT ALL CLASSES ALL FORMS     LINES 139        LINE 1-3 (3)
COMMAND INPUT ===> _                                      SCROLL ===> 1
PREFIX=L*  DEST=(ALL)  OWNER=*  SYSNAME=
NP   JOBNAME  JobID    Owner    Prty C Forms    Dest                Tot-Rec
     LOGINT   STC03961 IBMUSER     9 A STD      LOCAL                    47
     LOGINT   STC03963 IBMUSER     9 A STD      LOCAL                    50
     LOGMSG1  STC03951 IBMUSER     9 X STD      LOCAL                    42
```

Figure 1.20 Filtered job list in SDSF

To remove the filter, run this command:

```
PREFIX
```

1.4 z/OS UNIX System Services

Many changes have occurred in the world of computing since the announcement of System/360 in 1964. Among the many significant changes is the development of the UNIX operating system by employees at AT&T's Bell Labs in the 1960s. Although UNIX has concepts such as processes and threads, which are analogous to z/OS concepts such as address spaces and tasks, many significant differences exist. For example, in UNIX, files are byte-oriented streams of data, but in z/OS, files (data sets) are record oriented.

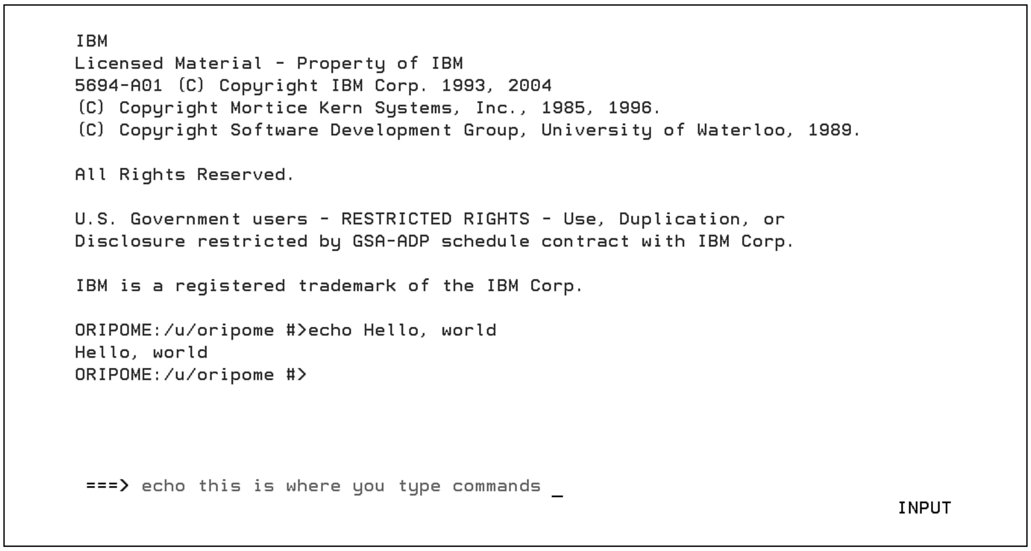
Within z/OS, you have a complete UNIX environment with z/OS UNIX System Services. This UNIX environment is integrated with the "traditional" z/OS environment. For example, you can access a z/OS UNIX file from a batch job and you can access a data set from a z/OS UNIX application.

You can enter the world of UNIX from z/OS in several ways. From TSO, you can enter the z/OS UNIX environment using the `OMVS` command from the TSO `READY` prompt. Within the ISPF environment, you can type the command **`tso omvs`** to enter UNIX (in general, you can run any TSO command from ISPF by prefacing it with `tso`).

Why OMVS?

When IBM first added the UNIX environment to MVS, it was called Open Edition, with the "Open" designating this environment and set of interfaces as one that was not designed or owned by IBM. The logical extension of this is Open MVS, which was shortened to the command `OMVS`. You can see vestiges of this naming convention in the `OMVS` command, the OMVS segment in user profiles, and the z/OS UNIX "O" commands, such as `OGET`, `OPUT`, and `OEDIT`.

Regardless of the way you enter it, OMVS provides a shell interface where you can type UNIX commands, as shown in Figure 1.21. By default, you type commands close to the bottom, at the ===> prompt (it is possible to configure OMVS to place the ===> prompt at the top instead).



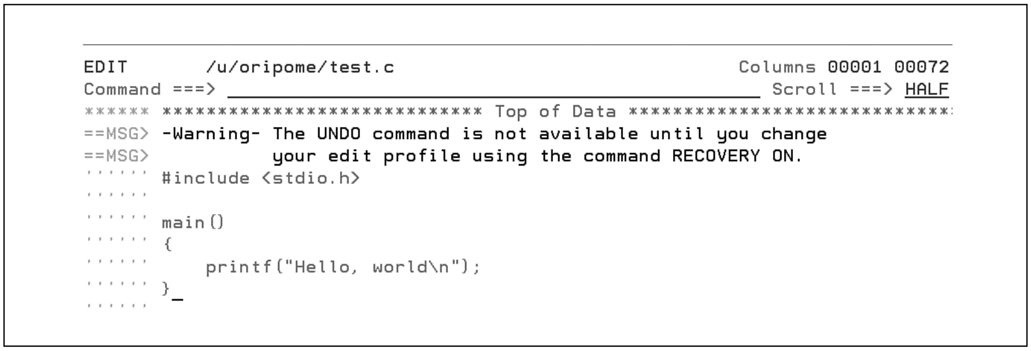
```
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2004
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

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.

ORIPOME:/u/oripome #>echo Hello, world
Hello, world
ORIPOME:/u/oripome #>



 ===> echo this is where you type commands _
                                                                    INPUT
```

Figure 1.21 OMVS shell

Can we do our "Hello, World" example as a z/OS UNIX program? Sure! Let's do it in the C programming language. Start by using OEDIT, which is the ISPF editor for z/OS UNIX:

```
oedit test.c
```

You are now in an editor that is very similar to the editor that you used to edit your JCL. This time, enter the "Hello, World" program, as shown in Figure 1.22. When you are done, press **F3** to exit the editor. If it asks about log file disposition, as in Figure 1.15, enter **2** to delete it.



```
EDIT       /u/oripome/test.c                               Columns 00001 00072
Command ===> ____________________________________________ Scroll ===> HALF
****** ***************************** Top of Data *****************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' #include <stdio.h>
''''''
'''''' main()
'''''' {
''''''     printf("Hello, world\n");
'''''' }_
''''''
```

Figure 1.22 “Hello, World” program in C for OMVS

Now compile and execute the program:

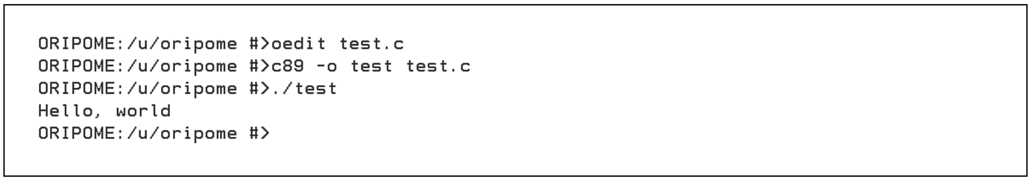
```
c89 -o test test.c
./test
```

It should write the message, as shown in Figure 1.23. If the compiler fails, type **`exit`** from OMVS and **`LOGOFF`** from TSO. Then when you log back on, type **`2096128`** in the size field of the TSO logon panel, the panel shown in Figure 1.7 earlier. Note that the C compiler is a separate product from OMVS, and your site might not have it.

The TSO Size Field

The size field in the TSO logon screen specifies the region size, the maximum amount of memory that will be allocated to you for that TSO logon. The C compiler requires a lot of memory, so you might need to increase this number from the default to about 2MB.

In JCL jobs, you can use `REGION=<number>` to specify the same information.



```
ORIPOME:/u/oripome #>oedit test.c
ORIPOME:/u/oripome #>c89 -o test test.c
ORIPOME:/u/oripome #>./test
Hello, world
ORIPOME:/u/oripome #>
```

Figure 1.23 Execution of the “Hello, World” program in C for OMVS

When you are done with OMVS, use **`exit`** to leave it.

1.5 Getting Help

At this point, we would like to write that you are completely comfortable with the mainframe's user interface and that you know how to do everything you might need to learn RACF. However, that would be a bold-faced lie. We could have filled the entire book with directions on how to use the mainframe and still not included everything you might need.

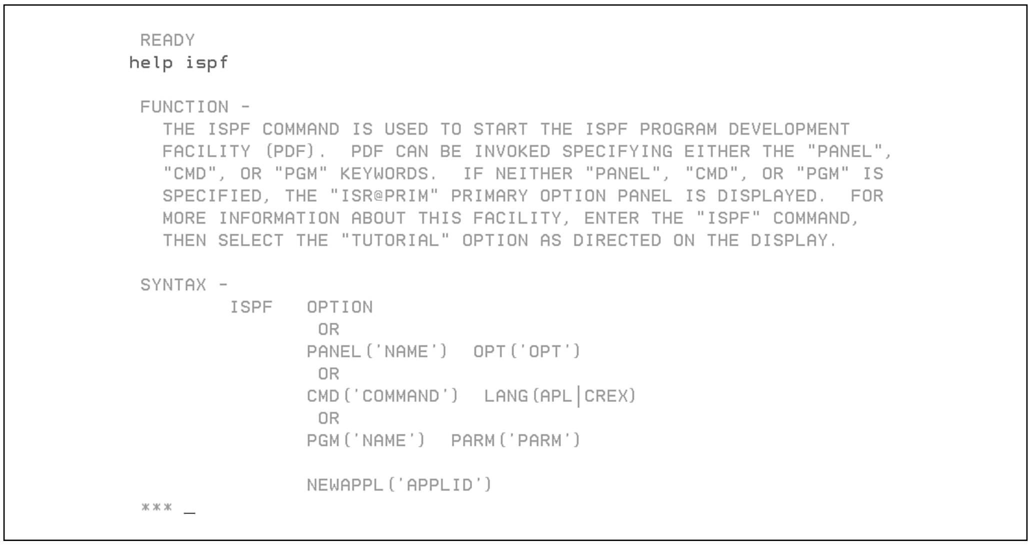
Instead, this section teaches you how to get help when you need it.

1.5.1 Context-Sensitive Help

The mainframe itself has a lot of documentation for your use.

1.5.1.1 TSO

To get help on a TSO command, type **`HELP <name of command>`**. For example, Figure 1.24 shows the help text for the `ISPF` command. The prompt at the bottom `(***)` shows that more information is available if you press Enter.



```
 READY
help ispf

 FUNCTION -
   THE ISPF COMMAND IS USED TO START THE ISPF PROGRAM DEVELOPMENT
   FACILITY (PDF).  PDF CAN BE INVOKED SPECIFYING EITHER THE "PANEL",
   "CMD", OR "PGM" KEYWORDS.  IF NEITHER "PANEL", "CMD", OR "PGM" IS
   SPECIFIED, THE "ISR@PRIM" PRIMARY OPTION PANEL IS DISPLAYED.  FOR
   MORE INFORMATION ABOUT THIS FACILITY, ENTER THE "ISPF" COMMAND,
   THEN SELECT THE "TUTORIAL" OPTION AS DIRECTED ON THE DISPLAY.

 SYNTAX -
         ISPF   OPTION
                 OR
                PANEL('NAME')  OPT('OPT')
                 OR
                CMD('COMMAND')  LANG(APL|CREX)
                 OR
                PGM('NAME')  PARM('PARM')

                NEWAPPL('APPLID')
 *** _
```

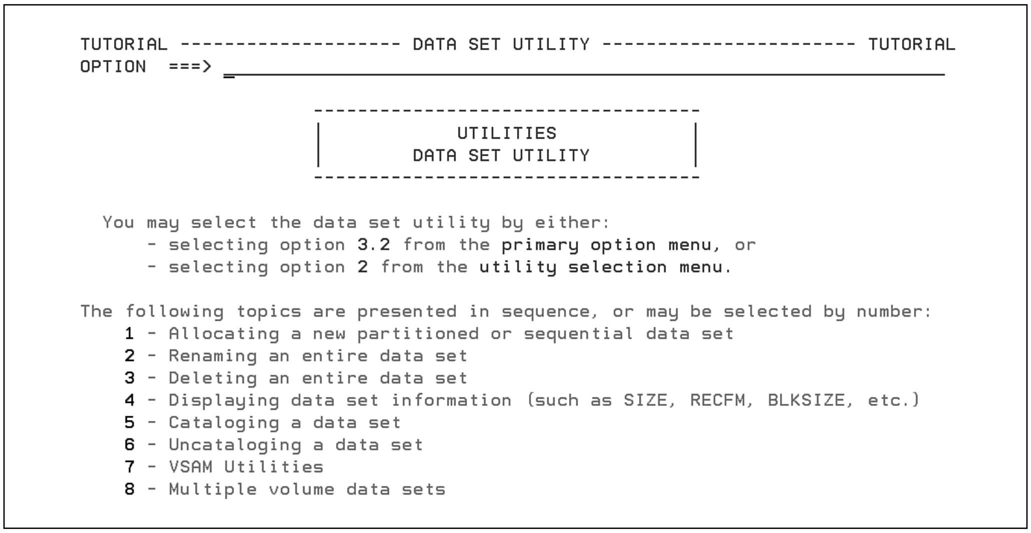
Figure 1.24 Output of the TSO HELP command

This help file shows that you can run ISPF with an option to immediately reach whatever panel you need. Use this to get to the ISPF data set utility panel, which is option 3 from the main menu followed by option 2 from the Utilities menu.

```
ISPF 3.2
```

1.5.1.2 ISPF

Inside ISPF, you can press F1 for context-sensitive help. For example, Figure 1.25 shows the context-sensitive help for the data set utilities panel. Inside the help panel, you can press **Enter** to advance to the next screen, or choose an option from the menu (if available). To get back to the panel, press **F3**.



```
TUTORIAL -------------------- DATA SET UTILITY ----------------------- TUTORIAL
OPTION  ===> _

                    -----------------------------------
                   |              UTILITIES            |
                   |          DATA SET UTILITY         |
                    -----------------------------------

  You may select the data set utility by either:
      - selecting option 3.2 from the primary option menu, or
      - selecting option 2 from the utility selection menu.

The following topics are presented in sequence, or may be selected by number:
    1 - Allocating a new partitioned or sequential data set
    2 - Renaming an entire data set
    3 - Deleting an entire data set
    4 - Displaying data set information (such as SIZE, RECFM, BLKSIZE, etc.)
    5 - Cataloging a data set
    6 - Uncataloging a data set
    7 - VSAM Utilities
    8 - Multiple volume data sets
```

Figure 1.25 ISPF context-sensitive help for the data set utilities panel

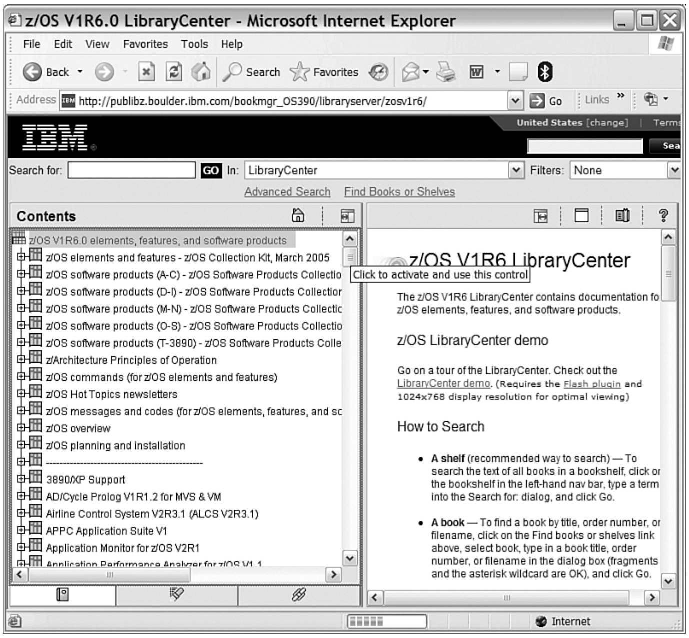
1.5.1.3 OMVS

In OMVS, you can press **F1** to get help about the shell itself. To get help for a specific command, use `man <name of command>`, as you would in any other version of UNIX.

1.5.2. The Manuals

IBM makes the manuals for z/OS available at http://publibz.boulder.ibm.com/bookmgr_OS390/libraryserver. Select your z/OS version and you will see a list of manuals, as shown in Figure 1.26.

For example, Figure 1.27 shows part of *z/OS 1R5.0-1R6.0 TSO/E User's Guide*, Topic 1.1.2.1, "Issuing the `LOGON` Command." This manual, similar to most manuals we will use in this book, is available under z/OS Elements and Features—z/OS Collection Kit (the top subdirectory in the list of manuals).


z/OS V1R6.0 LibraryCenter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://publibz.boulder.ibm.com/bookmgr_OS390/libraryserver/zosv1r6/
Go Links
IBM
United States [change] | Terms
Search for: GO In: LibraryCenter Filters: None
Advanced Search Find Books or Shelves
Contents
Click to activate and use this control
Internet

z/OS V1R6.0 elements, features, and software products
z/OS elements and features - z/OS Collection Kit, March 2005
z/OS software products (A-C) - z/OS Software Products Collectio
z/OS software products (D-I) - z/OS Software Products Collection
z/OS software products (M-N) - z/OS Software Products Collectio
z/OS software products (O-S) - z/OS Software Products Collectio
z/OS software products (T-3890) - z/OS Software Products Colle
z/Architecture Principles of Operation
z/OS commands (for z/OS elements and features)
z/OS Hot Topics newsletters
z/OS messages and codes (for z/OS elements, features, and sc
z/OS overview
z/OS planning and installation

3890/XP Support
AD/Cycle Prolog V1R1.2 for MVS & VM
Airline Control System V2R3.1 (ALCS V2R3.1)
APPC Application Suite V1
Application Monitor for z/OS V2R1
Application Performance Analyzer for z/OS V1.1

z/OS V1R6 LibraryCenter

The z/OS V1R6 LibraryCenter contains documentation fo
z/OS elements, features, and software products.

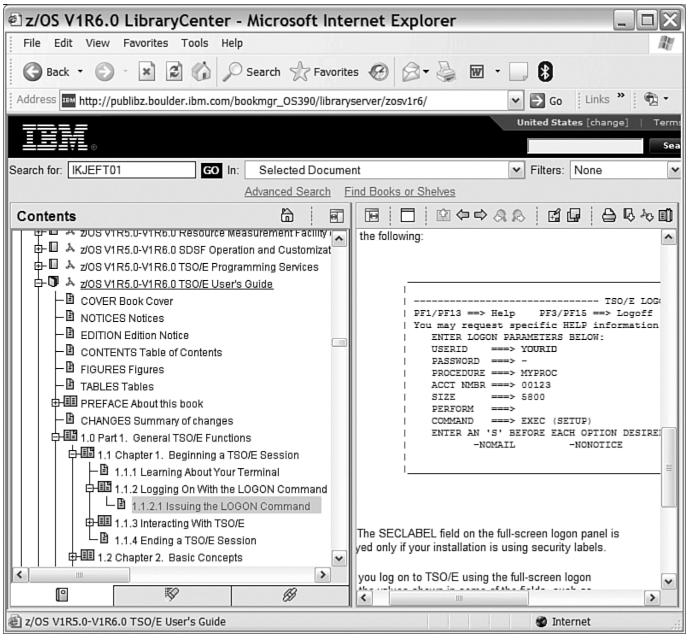
z/OS LibraryCenter demo

Go on a tour of the LibraryCenter. Check out the LibraryCenter demo. (Requires the Flash plugin and 1024x768 display resolution for optimal viewing)

How to Search

- **A shelf** (recommended way to search) — To search the text of all books in a bookshelf, click or the bookshelf in the left-hand nav bar, type a term into the Search for: dialog, and click Go.
- **A book** — To find a book by title, order number, or filename, click on the Find books or shelves link above, select book, type in a book title, order number, or filename in the dialog box (fragments and the asterisk wildcard are OK), and click Go.

Figure 1.26 List of manuals for z/OS 1.6


z/OS V1R6.0 LibraryCenter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://publibz.boulder.ibm.com/bookmgr_OS390/libraryserver/zosv1r6/
Go Links
IBM
United States [change] | Term
Search for: IKJEFT01 GO In: Selected Document Filters: None
Advanced Search Find Books or Shelves
Contents
z/OS V1R5.0-V1R6.0 Resource Measurement Facility
z/OS V1R5.0-V1R6.0 SDSF Operation and Customizat
z/OS V1R5.0-V1R6.0 TSO/E Programming Services
z/OS V1R5.0-V1R6.0 TSO/E User's Guide
COVER Book Cover
NOTICES Notices
EDITION Edition Notice
CONTENTS Table of Contents
FIGURES Figures
TABLES Tables
PREFACE About this book
CHANGES Summary of changes
1.0 Part 1. General TSO/E Functions
1.1 Chapter 1. Beginning a TSO/E Session
1.1.1 Learning About Your Terminal
1.1.2 Logging On With the LOGON Command
1.1.2.1 Issuing the LOGON Command
1.1.3 Interacting With TSO/E
1.1.4 Ending a TSO/E Session
1.2 Chapter 2. Basic Concepts
z/OS V1R5.0-V1R6.0 TSO/E User's Guide
Internet

the following:

```
| ------------------------------- TSO/E LOG
| PF1/PF13 ==> Help    PF3/PF15 ==> Logoff
| You may request specific HELP information
|    ENTER LOGON PARAMETERS BELOW:
|    USERID    ===> YOURID
|    PASSWORD  ===> -
|    PROCEDURE ===> MYPROC
|    ACCT NMBR ===> 00123
|    SIZE      ===> 5800
|    PERFORM   ===>
|    COMMAND   ===> EXEC (SETUP)
|    ENTER AN 'S' BEFORE EACH OPTION DESIRE
|           -NOMAIL          -NONOTICE
|
```

The SECLABEL field on the full-screen logon panel is
yed only if your installation is using security labels.

you log on to TSO/E using the full-screen logon

Figure 1.27 Example of a topic in a z/OS manual

1.6 Additional Information

A plethora of additional information is available on mainframes. Some of our favorite references are

- The IBM Redbooks® web site. Redbooks are books written by technical professionals working with the IBM International Technical Support Organization (ITSO). These books (and shorter works called Redpieces) are tactical "how to" books on a variety of subjects. The books and Redpieces "Introduction to the New Mainframe: z/OS Basics" and "Introduction to the New Mainframe: Networking" are available for free from the IBM Redbooks web site at www.redbooks.ibm.com. We highly recommend the books *Introduction to the New Mainframe: z/OS Basics*, *Introduction to the New Mainframe: Networking*, *Introduction to the New Mainframe: Security*, and *C/C++ Applications on z/OS and OS/390 UNIX.*
- The IBM publications web site for z/OS at www.ibm.com/servers/eserver/zseries/zos/bkserv/. From this web site, you can find almost all the product publications for the various z/OS products.
- z/OS ISPF Edit and Edit Macros, which explains the ISPF editor. It is available on the IBM publication web site for z/OS.

- *IBM's 360 and Early 370 Systems,* by Emerson W. Pugh, Lyle R. Johnson, and John H. Palmer (MIT Press, 1991), the definitive history of the development of the System/360.
- The IBM Archives, which contain a wealth of history of the mainframe, at www.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro.html.