Charting the Course...

...to Your Success!

ISPF and JCL on z/OS

ISPF and JCL on z/OS - Course Objectives

On successful completion of this class, the student, with the aid of the appropriate reference materials, should be able to:

1. Use full-screen terminals, including the appropriate Function keys, to accomplish work under ISPF/PDF
2. Use the CUA interface (action bars, pull-downs, point-and-shoot fields, etc.), and tailor the look and feel of ISPF to meet individual preferences
3. Describe the characteristics of, and differences between, sequential data sets, partitioned data sets (PDSs), and PDSEs (Partitioned Data Set Extended)
4. View a sequential data set or a member of a PDS/PDSE
5. Allocate, rename, and delete data sets or members, and print or display the attributes or contents of a data set
6. Copy and move data sets and members
7. Use productivity features such as command stacking and split screen processing, the CMDE command and command retrieval techniques
8. Edit data sets or members: create new members or files, and modify existing members or files
9. Understand the basic flow of work in z/OS, including JES Readers, Writers, Initiators, the role of the Interpreter, and the purpose of Allocation
10. Code JCL statements as necessary to accomplish work in the z/OS environment, including JOB, EXEC, DD, OUTPUT, IF/THEN, ELSE, ENDIF, INCLUDE, SET, JCLLIB, PROC and PEND statements
11. Copy files for backup, restore, and testing purposes using the IBM utility IEBGENER and use some basic services of IDCAMS, the VSAM utility
12. Use a Sort/Merge program product to sort a sequential data set
13. Use ISPF/PDF 3.8 and / or SDSF, Flasher, IOF, or (E)JES facilities for tracking jobs and examining job output
14. Code cataloged procedures, including the use of symbolic parameters and defaults, nested procedures, and private proclibs
15. Describe the implications of Storage Management Subsystem (SMS) and Partitioned Data Sets, Extended (PDSE's).

ISPF and JCL on z/OS - Topical Outline

Day One

ISPF and JCL on z/OS - Topical Outline, p.2.

Day Two

ISPF and JCL on z/OS - Topical Outline, p.3.

Day Three

Day Four

Day Five

Section Preview

❒ **ISPF Introduction**

- ♦ **TSO / ISPF / PDF**
- ♦ **Keyboard Notes**
- ♦ **The Logon Process**
- ♦ **ISPF/PDF Primary Option Menu**
- ♦ **Standard Panel Format**
- ♦ **CUA Panel Formats**
- ♦ **Using Action Bars**
- ♦ **Getting Around in ISPF**
- ♦ **Leaving ISPF**
- ♦ **A First Encounter With ISPF / PDF (Machine Exercise)**

TSO / ISPF / PDF

TSO - Time Sharing Option

A powerful but awkward-to-use facility that allows users at any kind of terminal to:

- ❒ **Create, modify, delete, rename files**
- ❒ **Maintain libraries of programs, JCL, data**
- ❒ **Assemble, compile, link, run programs in the batch (background) or under immediate terminal control (foreground)**
- ❒ **Monitor status of batch jobs, examine output**
- ❒ **Communicate to operator or other users**

ISPF - Interactive System Productivity Facility

An extension to TSO, for users of full-screen terminals, that vastly simplifies using TSO by providing:

- ❒ **Support for creating and using screens (<u>panels</u>) to gather, present, and modify data**
- ❒ **Support for creating and displaying embedded HELP and tutorial information**
- ❒ **Support for using programmable function (PF) keys**
- ❒ **Interfaces to programming languages such as CLIST, REXX, COBOL, Assembler, and FORTRAN**

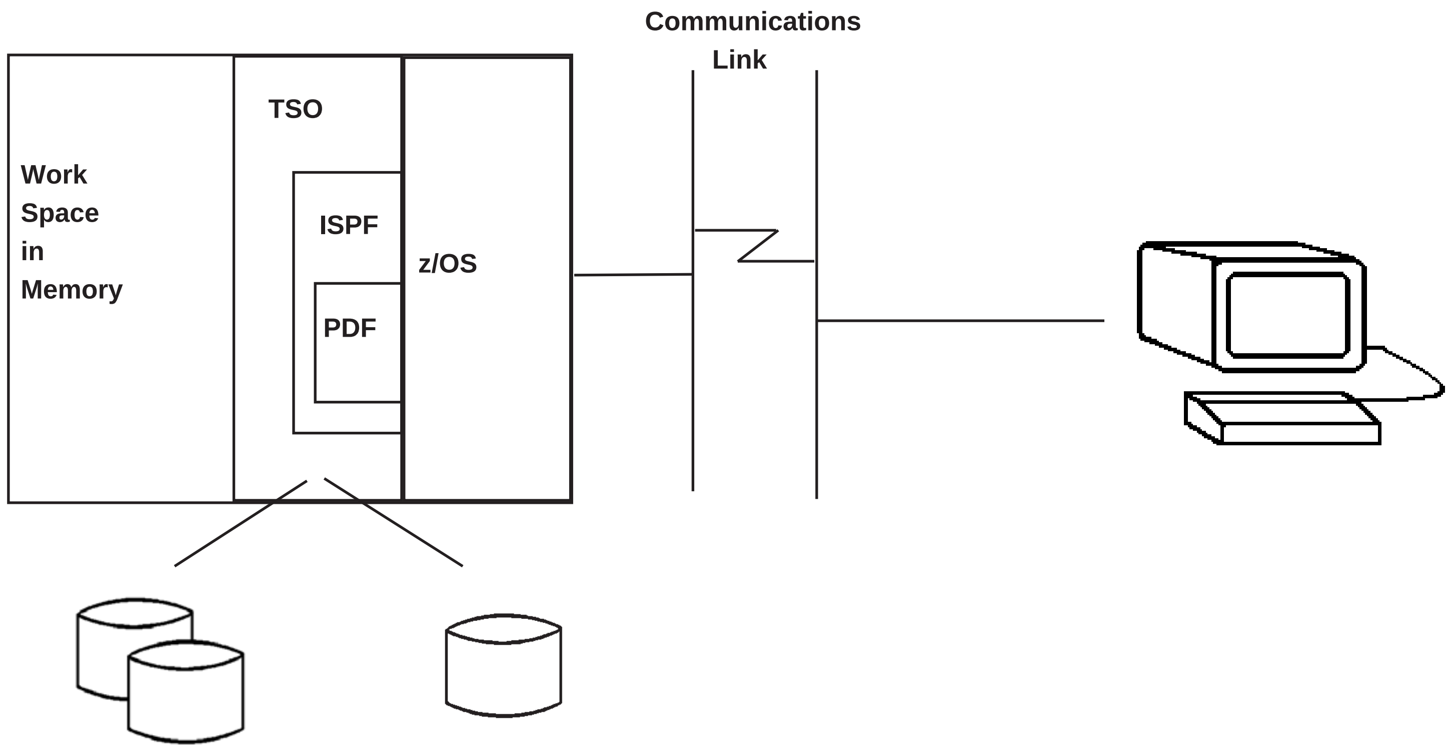
ISPF Is a <u>Dialog Manager</u>

PDF - Program Development Facility

A dialog that runs under ISPF that provides programmers assistance in using TSO through:

- ❒ **Menu and fill-in-the-blanks approach**
- ❒ **Full screen editing and browsing of programs and data**

The TSO / ISPF Environment



Files / Libraries / Job Queue
Temporary Work Space

Keyboard Notes

❒ **Every keyboard is different, it seems, these days**

- ♦ **Yet it's important to learn certain special functions and then each time you sit down to a new keyboard to learn what keys to use to obtain the functions**

❒ **In addition to the standard alphanumeric and punctuation keys, ISPF takes advantage of these keys, if available:**

- ♦ **<u>Function Keys</u> — assigned by the system or the user to command combinations, so a single keystroke can issue one or more commands**

  - ✗ Most keyboards come with 12 or 24 function keys

  - ✗ Also called PF keys in some environments (Programmable Function keys)

- ♦ **<u>Insert</u> — insert key; puts you in insert mode for keying in data in the middle of a line**

- ♦ **<u>Delete</u> — delete one character and close up line from the right**

- ♦ **<u>Reset</u> — unlock keyboard when it locks up**

- ♦ **<u>Erase EOF</u> — erase to End of Field; when pressed, all characters to the right of the cursor are erased**

- ♦ **<u>Home</u> — sends cursor to first input field on the screen**

Keyboard Notes, 2

❒ **Additional keys to know**

- ♦ **Attn — attention; interrupts a long-running process**
- ♦ **PA1 — Program Attention 1; use in place of Attn if not present**
- ♦ **PA2 — Reshow; erases input from screen that has not yet been transmitted to the host**
- ♦ **Enter — transmit screen contents back to host**
- ♦ **New Line — move cursor to next line, do not transmit to host**
- ♦ **Arrow keys — move cursor on screen in direction of arrow**
- ♦ **Tab — tab cursor to next input location (Shift+Tab will tab to previous input location)**

❒ **Take a few minutes now and learn what keys perform these functions on the keyboard you will be using**

- ♦ **If you are using a PC for a terminal, you are using what's called emulator software: programs in the PC that make the PC behave like a standard mainframe terminal**
- ♦ **Some of the available emulators let you choose what keys on the PC to use for these mainframe keyboard functions; see if you can figure out how to change keyboard mappings**

The Logon Process

❒ **To get into ISPF, which will be our primary tool during this class, you need to accomplish these steps**

- ♦ **Get onto the network**
- ♦ **Logon to the correct machine (this puts you into TSO)**
- ♦ **Get to ISPF**

❒ **Before you can do this, you must have been assigned a TSO user id (or, more commonly, just a TSO id) and a password**

- ♦ **The <u>TSO id</u> identifies you to TSO; more than one person can have the same name, but only one person can be logged onto the system using a given TSO id; TSO id's are unique to you for a given TSO system**
- ♦ **<u>Passwords</u> are also unique to you; if someone knows your TSO id they could get onto the system using your id**
  - ✗ This would keep you from logging in at the same time, and give this other person access to all authorities you may have
  - ✗ So other people may know your TSO id (for example, they may need to send messages to you, and you do this by TSO id), but **no one else should know your password**
  - ✗ In some companies, letting anyone else know your password is a firing offense; do not take this lightly

Passwords

- ❒ **Passwords are maintained by programs that maintain security, and various security programs have differing rules for making up passwords; here, the general rules are:**
  - ♦ **Length:**
  - ♦ **Composition:**
- ❒ **In addition, passwords automatically expire every __ days**
  - ♦ **This requires you to create a new password when you logon and your password has expired**
  - ♦ **And, since the security package keeps track of the last __ passwords you've used in addition to your current password, your new password needs to be a password you haven't used in a while**
- ❒ **If you forget your password, call your security administrator**
  - ♦ **After verifying you are who you say you are, they will give you a new password that is already set as expired**
    - ✗ Logon using the new password and then, since it has expired, the system will prompt you for a new password
    - ✗ This way, once again, not even the security administrator will know your password

Passwords, continued

- ❒ **If you are logging on and enter an incorrect password, after __ tries your TSO id is de-activated**

  - ♦ **It will take another call to your security administrator to have your TSO id re-activated and your password will be re-set (with an expired attribute)**

  - ♦ **This is done to prevent un-authorized personnel trying to get into the system using your TSO id and simply trying a large number of possible passwords**

- ❒ **Some other guidelines when making up passwords**

  - ♦ **<u>Don't</u> use words found in the dictionary (transliterate letters, append or insert numbers, etc.)**

  - ♦ **<u>Don't</u> use names: your name, your spouse's name, your children's names, your favorite city, etc.**

  - ♦ **<u>Don't</u> use a pattern easy to guess (e.g.: APRIL05, MAY05, ...)**

  - ♦ **<u>Do</u> use a string that's easy to remember but hard to guess**

- ❒ **Clearly it's important to know the name and phone number of your security administrator, along with your TSO id and current password**

- ❒ **In z/OS V1.10, <u>pass phrase</u>s were introduced, allowing a case sensitive phrase of up to 100 characters long instead of a password**

Do What I Say, Not What I Do

- ❐ **All of the above being said, we may soon assign training TSO id's for you to use in this class**
  - ♦ **If so, everyone will know your TSO id, and everyone will start out with the same (expired) password that follows a pattern**
  - ♦ **When you logon, change your password to some string only you know**
  - ♦ **After end of class, the TSO id's will be de-activated by the training people, for use by some later class**
    - ✗ Before you leave, we'll show you how to copy any files you want to keep from your training TSO id to your personal TSO id
- ❐ **On the other hand, you may simply use your own TSO id if you have one**

Back to the Logon Process

- ❒ These classes are being taught in a variety of environments using a range of terminals

  - ♦ For this reason, we have not included the early steps for logging on in the materials

  - ♦ At this point, take notes as the instructor walks you through the process of getting to the network logon screen:

Logon, continued

❐ **At this point, in some installations you'll be at the logon screen, whereas in others you may find you need to go through one or more intermediate screens**

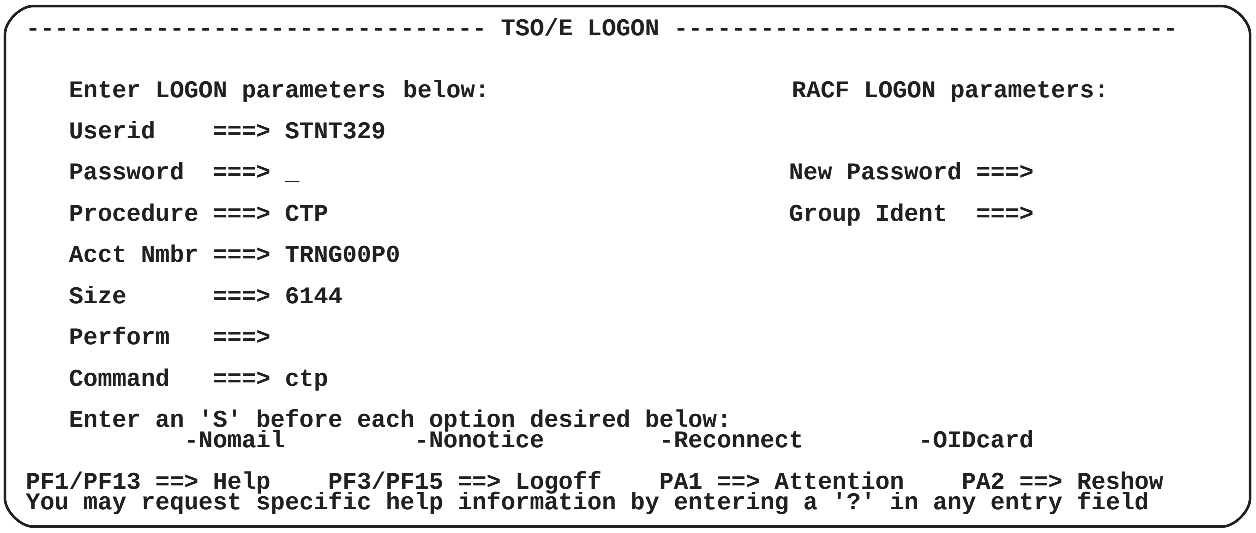
❐ **Notes:**

❐ **At some point, you'll Key in your TSO id and press Enter to see the next screen ...**

Logon Screen

❐ **This screen is where you enter your password**

- ♦ **And you can enter a new password to change your current password if you would like to or if you need to**



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

   Enter LOGON parameters below:                    RACF LOGON parameters:

   Userid    ===> STNT329

   Password  ===> _                                 New Password ===>

   Procedure ===> CTP                               Group Ident  ===>

   Acct Nmbr ===> TRNG00P0

   Size      ===> 6144

   Perform   ===>

   Command   ===> ctp

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

PF1/PF13 ==> Help    PF3/PF15 ==> Logoff    PA1 ==> Attention    PA2 ==> Reshow
You may request specific help information by entering a '?' in any entry field
```

❐ **Enter your password, and possibly change some of the other selections, and press Enter ...**

Logon Messages

❒ **After entering your password, you'll see any broadcast messages that are in effect, and possibly messages directed specifically to you**

- **The details vary from day to day; here's an example from a recent logon:**

```
ICH70001I STNT329  LAST ACCESS AT 08:28:27 ON TUESDAY, xxxxxxxx 21, 200x
STNT329 LOGON IN PROGRESS AT 08:32:30 ON xxxxxxxx 21, 200x
 * * * * * * * * * * * * * * * * * *

       THIS IS COTHOR   SY6D

 PLEASE CALL (nnn) 451-2108 FOR ASSISTANCE


Attn: BRUTUTIL removal has been changed to 2/28/yy. See TIB #39

 * * * * * * * * * * * * * * * * * *
********************************************************
EXECUTE CLIST 'CTP' TO ENTER COMMON LOGON SCREENS
IF DB2 IS TO BE INVOKED IN THIS SESSION, DB2 MUST BE
INDICATED WHEN INVOKING THE 'CTP' CLIST. ENTER 'DB2'
AS A PARM TO THE CLIST.
EX: CTP DB2
********************************************************
ctp
 IBM TIB 16 * IMS n.n - PRODUCT UPDATE I:02/20/yy L:02/20/yy
 IBM TIB 25 * REMOVAL OF PANVALET AND PANEXEC I:02/20/yy L:02/20/yy
 IBM TIB 24 * NEW BINDER REPLACES LINKAGEnI:02/20/yy L:02/20/yy
***
```

❒ **Notice the three asterisks on the last line**

- **In TSO, whenever you see a line of just three asterisks, the system is waiting for you to press Enter before going on**

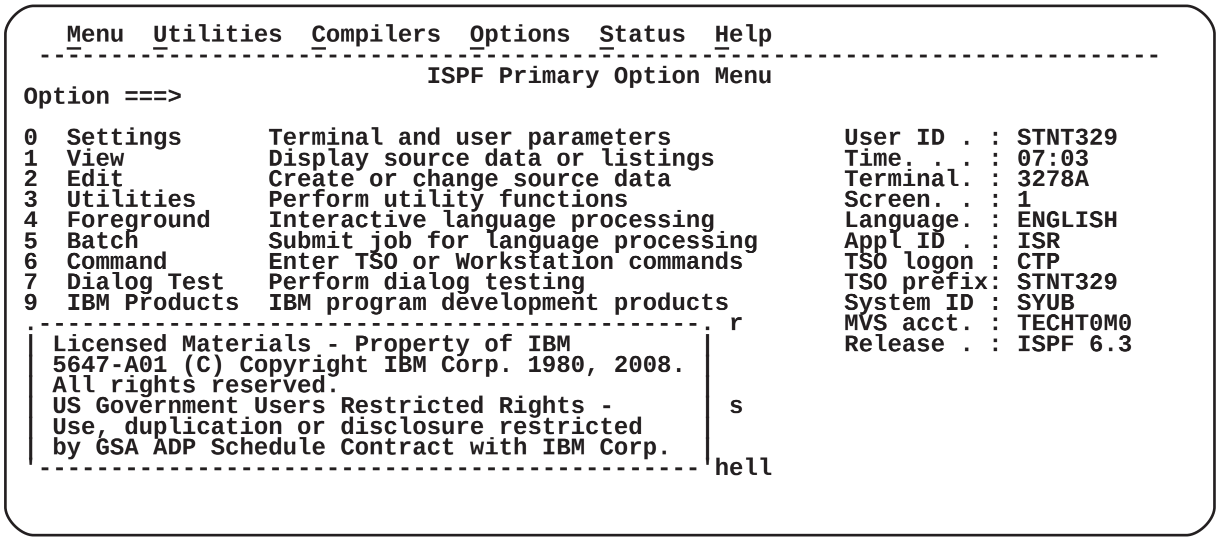
- **If the screen has entry fields, any keying you do before pressing Enter to clear the three asterisks will be ignored**

❒ **Sometimes you may have several screens of broadcast messages**

- **Just press Enter after checking each screen**

ISPF Primary Option Menu

❒ In some installations, again, you may have some intermediate levels, but usually by now we are at the main menu for ISPF, the Primary Option Menu:



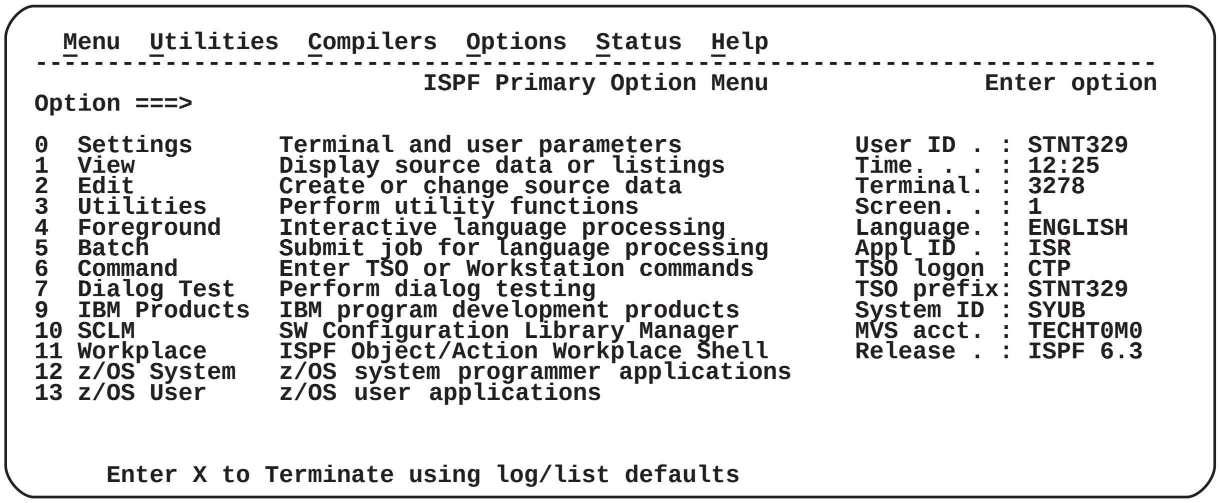
```
   Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                            ISPF Primary Option Menu
Option ===>

0  Settings      Terminal and user parameters           User ID . : STNT329
1  View          Display source data or listings        Time. . . : 07:03
2  Edit          Create or change source data           Terminal. : 3278A
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 : CTP
7  Dialog Test   Perform dialog testing                 TSO prefix: STNT329
9  IBM Products  IBM program development products       System ID : SYUB
 .-----------------------------------------------. r      MVS acct. : TECHT0M0
 | Licensed Materials - Property of IBM           |       Release . : ISPF 6.3
 | 5647-A01 (C) Copyright IBM Corp. 1980, 2008.   |
 | All rights reserved.                           |
 | US Government Users Restricted Rights -        | s
 | Use, duplication or disclosure restricted      |
 | by GSA ADP Schedule Contract with IBM Corp.    |
 '-----------------------------------------------'hell
```

❒ Note that part of the panel above is obscured by a copyright notice

- ♦ Press Enter to see the full screen ...

ISPF Primary Option Menu, 2

❒ **Finally, we see the complete main menu for ISPF, the Primary Option Menu:**



```
  Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                           ISPF Primary Option Menu               Enter option
Option ===>

0  Settings      Terminal and user parameters          User ID . : STNT329
1  View          Display source data or listings       Time. . . : 12:25
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 : CTP
7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
9  IBM Products  IBM program development products      System ID : SYUB
10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell    Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications


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

❒ **This is our real starting point for the work we do in this class**

- **In particular, we will focus on option 2 (Edit), but we will also work with options 0, 1, 3, and 6**

❒ **The Release level identifies the release of ISPF running and implies, roughly, the version of z/OS currently running**

✗ 5.2 -> z/OS V1.3, z/OS V1.4

✗ 5.5 -> z/OS 1.5

✗ 5.6 -> z/OS 1.6

✗ 5.7 -> z/OS 1.7

✗ 5.8 -> z/OS 1.8

✗ 5.9 -> z/OS 1.9

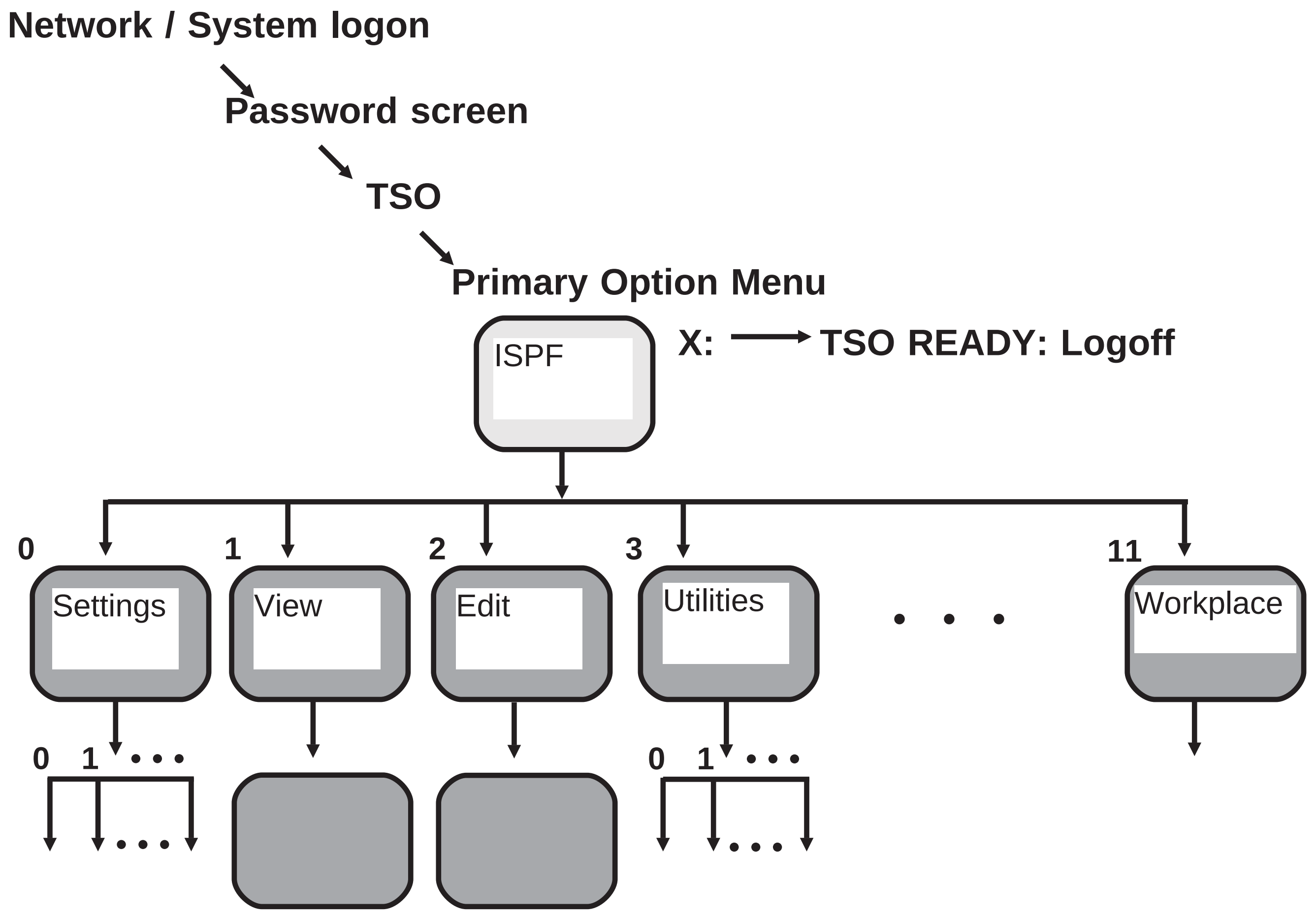
✗ 6.0 -> z/OS 1.10

✗ 6.1 -> z/OS 1.11, z/OS 1.12

✗ 6.3 -> z/OS 1.13

❒ **Just for a moment, let's step back and see how all the panels relate ...**

ISPF Panel Hierarchy



Panel Styles

❒ **There are a variety of panel styles in ISPF, for historical reasons**

- ♦ **Initially, ISPF panels displayed on 3270-style terminals, using what came to be called "Standard Format"**

  - ✗ We show an example of that later

- ♦ **Later, IBM introduced a standard called CUA (for Common User Access)**

  - ✗ Initially this was an attempt for a "dumb terminal" to look and feel a little bit like working with a PC / workstation using a GUI (Graphical User Interface) such as Windows or OS/2

  - ✗ To provide a consistency between the old, familiar, standard look and CUA, however, the user was given lots of flexibility to modify the look and feel

    - ➢ So an ISPF session could look like a standard session, or like a CUA session, or something in between, whatever the user preferred

  - ✗ If you are running on 3270-style terminal, or on a workstation that uses an emulator to act like a 3270-style terminal, this is what you will be seeing (and for these materials, our screen images will look like what you'll be working with)

Panel Styles, continued

- **If you are using a workstation, you can use the Client / Server (C/S) version of ISPF**

    - **This version uses the facilities of Windows, OS/2, AIX, or HP-UX to actually offload some of the processing to the workstation**

    - **An ISPF session running this way will look more like a traditional Windows, OS/2, AIX, or HP-UX application**

        - Using colors, graphics, push buttons, sizing, and other components like other applications on these platforms

    - **The literature also refers to using this facility as “running in GUI mode”**

- **Because of all this, screens / panels / windows look slightly different among the standard, CUA, and C/S modes**

    - **Especially since everyone can tailor the look to meet their own preferences**

    - **And occasionally, there will be small functional differences (which we will point out as appropriate)**

Standard Panel Format

❐ **The first three lines of most Standard format panels have special reserved functions, leaving the rest of the screen available for use by each specific application**

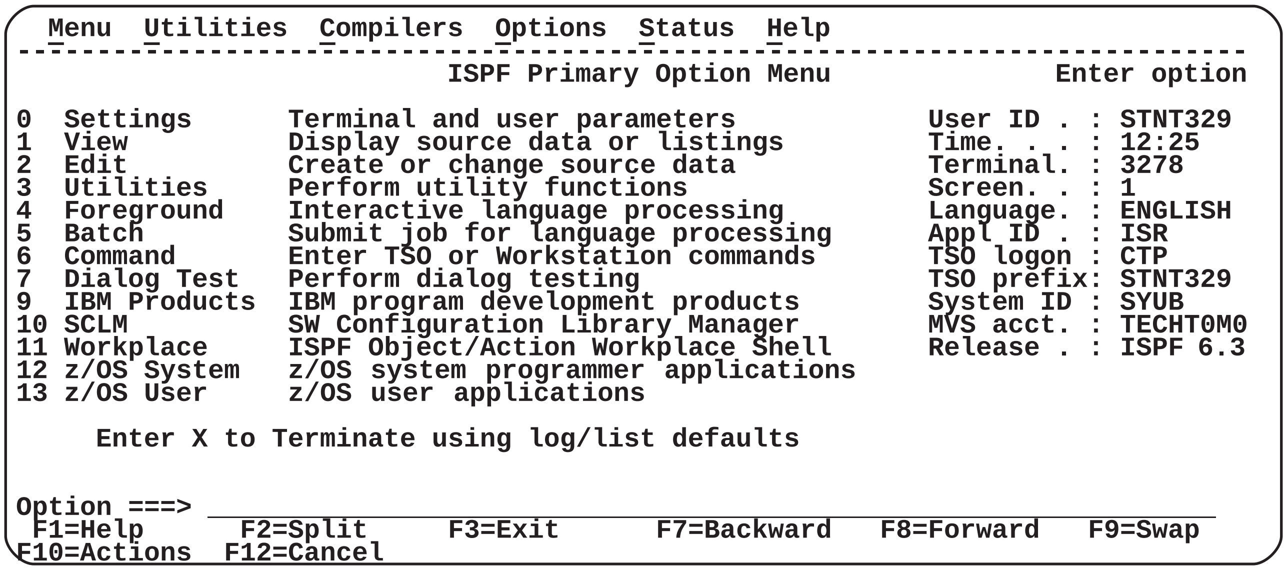
Line 1	title of panel	Short Message
Line 2	command / option line	
Line 3	Long Message	

- **The title line may include output from one of the screen labeling commands SCRNAME, SYSNAME, PANELID, USERID**
- **Short message area used to pass information about a request (status, error, etc.)**
- **Long message area normally used for headers, data, etc.**
    - ✗ But if an error message appears in the short message area, issuing a HELP command (or pressing the HELP Function key) will cause further explanation to display in the long message area
    - ✗ Issuing HELP again will take you into the tutorial to access the information available there

CUA Panel Formats

❒ **The standard CUA Panel Format has these components**

Action Bar
Separator Line
Panel Title / short message area
Long Message Area
Panel Body
Command / Option line
Function key area



```
  Menu  Utilities  Compilers  Options  Status  Help
----------------------------------------------------------------------------
                           ISPF Primary Option Menu              Enter option

0  Settings      Terminal and user parameters          User ID . : STNT329
1  View          Display source data or listings       Time. . . : 12:25
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 : CTP
7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
9  IBM Products  IBM program development products      System ID : SYUB
10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell    Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications

     Enter X to Terminate using log/list defaults


Option ===> ______________________________________________________________
 F1=Help      F2=Split     F3=Exit      F7=Backward   F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

❒ **Using the Settings choice, the command / option line can be moved below the title and the Function key display at the bottom can be removed**

- ♦ **Which is how we display panels in this course**

- ♦ **Other attributes of the look may be changed also**

Using Action Bars

❐ **The action bar contains a list of choices for you to consider**

- ♦ **To select one of these choices, first get to the action bar**

  ✗ Either use the arrow keys, or press F10, which is assigned to the command **Actions** (toggle between the Action bar and panel body)

  ➢ Alternatively, the Tab key will cycle you to the Action bar, and the Home key will also take you there

- ♦ **Use the Tab key or the right and left arrow keys to move along the action bar to the choice you are interested in**

- ♦ **Press Enter when the cursor is positioned on the choice you want**

  ✗ A pull-down menu appears, which is a numbered list of subchoices from the menu; for example, if you selected Options from the main panel, you'll see something like this:

```
  Menu  Utilities  Compilers  Options  Status  Help
 ---------------------------- .------------------------------. ----------------
                              |    1. General Settings        |
 Option ===>                  |    2. CUA Attributes...       |
                              |    3. Keylists...             |
 0  Settings      Terminal a  |    4. Point-and-Shoot...      |  ID . : STNT329
 1  View          Display so  |    5. Colors...               | . . . : 12:25
 2  Edit          Create or   |    6. Dialog Test appl ID...  | inal. : 3278
 3  Utilities     Perform ut  '------------------------------'  en. . : 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 : CTP
 7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
 9  IBM Products  IBM program development products      System ID : SYUB
 10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
 11 Workplace     ISPF Object/Action Workplace Shell    Release . : ISPF 6.3
 12 z/OS System   z/OS system programmer applications
 13 z/OS User     z/OS user applications

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

- ♦ **Select a choice by typing its number and pressing Enter**

Working With Pull-down Menus

- ❐ **If a pull-down menu choice has an asterisk (*) next to it, that indicates the choice is not currently available**
  - ♦ **Perhaps it is already in effect, for example**
  - ♦ **Note that in GUI (Client/Server) mode, unavailable choices are greyed**

- ❐ **If a pull-down menu choice has an ellipsis (...) next to it, that indicates that selecting that choice will cause a pop-up window to appear**
  - ♦ **For example, the Colors... choice on the previous page causes a panel to appear that allows you to change how colors are used for your session (if your terminal / emulator supports that)**

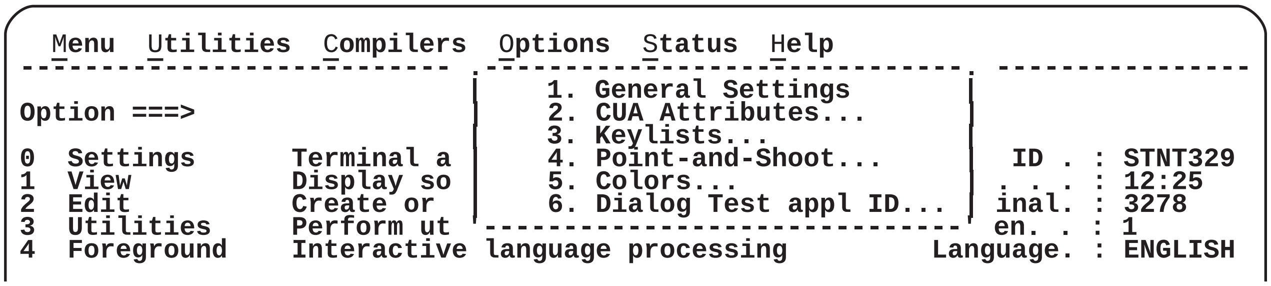
- ❐ **If a pull-down menu choice has neither an asterisk nor an ellipsis, selecting that choice performs the related action immediately**
  - ♦ **This may involve simply changing an attribute, invoking a new function, or taking any action that does not require additional information or confirmation from you**

- ❐ **If you want to remove a pull-down menu without selecting any choice, press F12 (Cancel)**

Menu Mnemonics

❒ **ISPF supports <u>mnemonics</u> in the menu (Action bar) choices**

- ♦ **A mnemonic appears as an underlined character in an Action bar, for example:**



```
  Menu  Utilities  Compilers  Options  Status  Help
----------------------------- .-----------------------------. ----------------
                              |    1. General Settings       |
Option ===>                   |    2. CUA Attributes...      |
                              |    3. Keylists...            |
0  Settings      Terminal a   |    4. Point-and-Shoot...     |  ID . : STNT329
1  View          Display so   |    5. Colors...              | . . . : 12:25
2  Edit          Create or    |    6. Dialog Test appl ID... | inal. : 3278
3  Utilities     Perform ut   '-----------------------------' en. . : 1
4  Foreground    Interactive language processing        Language. : ENGLISH
```

- ♦ **Keying in "Actions** x**", where** x **is the mnemonic for the action, selects that action immediately**

  - ✗ If you assign <u>Actions</u> to a function key, then you can type in the mnemonic and press the function key

❒ **The net effect is to speed the process of getting work done wherever mnemonics are implemented**

Getting Around in ISPF

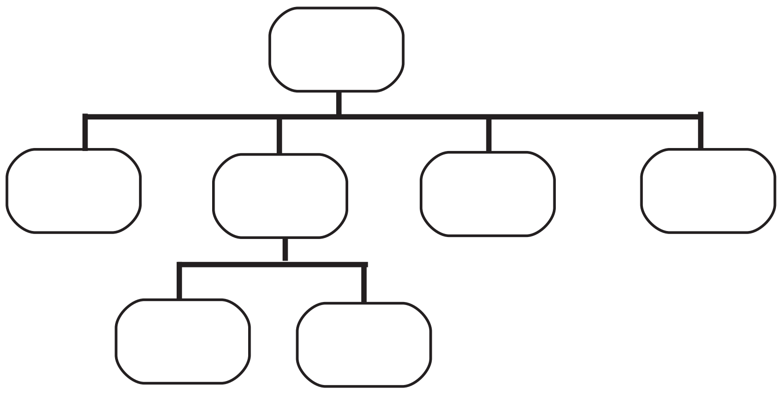
- **To select an option in ISPF, there are usually several routes**
  - **From a menu, key in the option's corresponding number**
  - **From the command / option line issue a command**
    - Alternatively, press a Function key that has been assigned to the command
  - **From the Action bar, select a choice then select a subchoice from the resulting pull-down menu**
    - If that choice is a new function (for example, if you are currently using Utilities and select Edit), your current process (Utilities) is suspended and the new process (Edit) is begun
      - Use the <u>Exit</u> command to terminate the current function (Edit) and return to the previous function (Utilities)
      - Use <u>End</u> to back up one level in the hierarchy of ISPF
      - Use <u>Cancel</u> to cancel a pull-down menu and return the cursor to the first action bar choice

- **When you are in a panel or window where you fill in the blanks, pressing Enter accepts the data / choices and moves forward into the application**
  - **Pressing F3 (Exit or End, depending on where you are) backs out of the panel without performing any action**

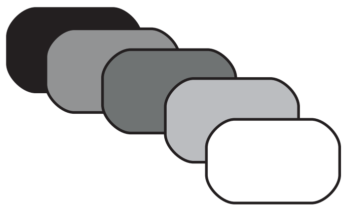
Getting Around in ISPF, 2

❒ **Notice the difference in using the hierarchy (using menus and options) versus nesting functions (using the Action bar)**

- ♦ **Using the hierarchy, you generally go up and down in an orderly fashion**



- ♦ **Using nesting (the Action bar), you stack functions one on top of another**



ISPF Commands

❒ **Some commands that can be entered from the command / option line on any screen:**

Help

♦ **Request help with current screen**

✗ HELP is context sensitive; that is, you will get help for the current panel you are on, or sometimes help for the current message, or help for the field where the cursor is located

Cancel

♦ **From a pull-down menu, the pull-down is removed and the cursor is placed on the first action bar choice**

♦ **Otherwise, the command is ignored**

Exit

♦ **Terminate the current function and return to the previous function, if any**

End

♦ **Leave current screen, back up one level higher in screen hierarchy**

Return

♦ **Back up immediately to the Primary Option Menu**

Getting Around in ISPF, concluded

- ❒ **From any menu, you can select an option to go forward to that option**
    - ♦ **From any screen, you can enter the END command to go back up one level in the hierarchy**
        - ✗ This is usually set to F3 and / or F15 (or Shift+F3 or Alt+F3 or Ctrl+F3, depending on your platform)
    - ♦ **From any screen, you can enter the RETURN command to go back to the top of the hierarchy**
        - ✗ This is often set to F4 and / or F16 (or Shift+F4 or Alt+F4 or Ctrl+F4, depending)
- ❒ **From any panel, you can get to the Action bar (Actions, or F10) and select a choice, resulting in a pull-down menu**
    - ♦ **From the Action bar, CANCEL (F12) cancels the pull-down**
    - ♦ **Selecting a choice from a pull-down may put you in a function**
        - ✗ EXIT (F3) returns you to the previous function; if there is no previous function, EXIT works the same as END
- ❒ **From the ISPF Primary Options Menu, option X gets you to the TSO READY prompt**
        - ✗ All set for you to logoff TSO ...

The LOGOFF Comand

Syntax

Logoff

Function

- **Terminates your TSO session, returns you to the network logon screen**

Computer Exercise: A First Encounter With ISPF/PDF

If you have not yet been given your TSO id and password, the instructor will assign them now. Then sit down at a terminal and go through the following steps

1. Logon to TSO and get into ISPF/PDF

2. Select each ISPF option from the menu, to peek at the next screen for each option. Back up from each screen using END (PF3 or F3).

3. Select each Action bar choice, to see the resulting menu; use Cancel (F12) to back out of the pull-down menus.

4. Experiment with Settings (either option 0 from the Primary Option Menu, option 1 from the Menu Action bar pull-down menu, or a command (Settings) from the command / option line).

   Move the command line to the bottom, and back to the top. Leave it wherever you like most.

   From the Function keys Action bar pull-down menu on the Settings window, set the function keys to show, then remove the function key display. Leave that setting wherever you like most.

5. Logoff the system.

At each point, make a note of any questions or observations, for sharing with the group after the lab.

Section Preview

❒ **ISPF Look and Feel**

- ♦ **The Settings Panel**
- ♦ **LIST and LOG Data Sets**
- ♦ **Function Keys Settings**
- ♦ **Working With Keylists**
- ♦ **Color, Intensity, and Highlighting**
- ♦ **Look and Feel: Options**
- ♦ **Look and Feel: Status**
- ♦ **Changing the Look and Feel of ISPF (Machine Exercise)**

ISPF Look and Feel

❒ **There are three places where you can change the look and feel of the ISPF panels / windows you work with**

- ♦ **<u>Settings</u> - specify terminal characteristics, screen layout, some processing characteristics; can change colors and keylists; can specify GUI characteristics**
- ♦ **<u>Options</u> - can invoke Settings; can specify CUA attributes and keylists, can change colors and establish point-and-shoot parameters**
- ♦ **<u>Status</u> - specify the contents of the Primary Option Menu's status area**

❒ **There is some overlap and redundancy here**

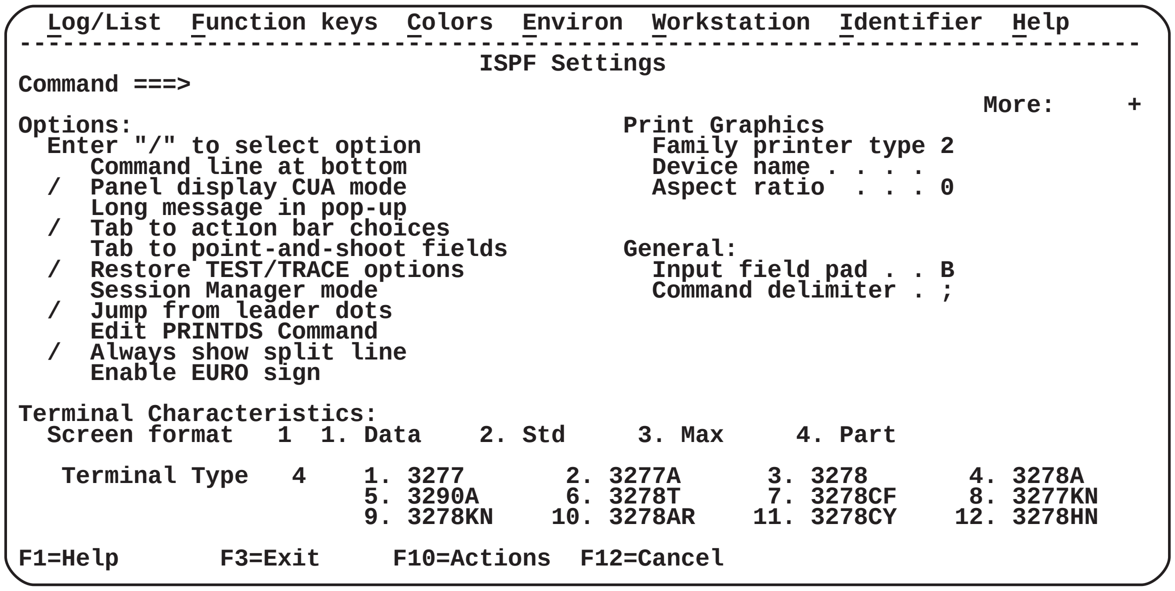
- ♦ **We expect each person will find their own preferred way to change the look and feel characteristics**
- ♦ **Here we will discuss the most useful attributes / options**

❒ **You get to the Settings panel from the Primary Option Menu in one of three ways**

- ♦ **Select Option 0 from the list**
- ♦ **Enter the "Settings" command on the option line**
- ♦ **Select option 1 from the Menu pull-down**

❒ **In any case, you will see the Settings panel ...**

The Settings Panel



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
------------------------------------------------------------------------------
                                ISPF Settings
Command ===>
                                                                 More:     +
Options:                                  Print Graphics
  Enter "/" to select option                Family printer type 2
     Command line at bottom                 Device name . . . .
  /  Panel display CUA mode                 Aspect ratio  . . . 0
     Long message in pop-up
  /  Tab to action bar choices
     Tab to point-and-shoot fields        General:
  /  Restore TEST/TRACE options             Input field pad . . B
     Session Manager mode                   Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics:
  Screen format   1  1. Data    2. Std     3. Max     4. Part

   Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                        5. 3290A      6. 3278T      7. 3278CF     8. 3277KN
                        9. 3278KN    10. 3278AR    11. 3278CY    12. 3278HN

F1=Help       F3=Exit     F10=Actions  F12=Cancel
```

Notes

♦ **Under Options, key a slash (/) to select an option, a space (blank) to deselect an option (actually, any non-blank character will work the same as the slash)**

♦ **We ignore the Print Graphics section as beyond the scope of this course**

♦ **We ignore the details on Screen format and Terminal Type: specify what hardware you are using (or emulating)**

✗ You must be given the appropriate information (although you can freely experiment without hurting anything)

♦ **We discuss the Options choices and the General choices on the following pages ...**

Settings Possibilites

❒ **Three related settings ...**

- ♦ **<u>Command line at bottom</u> - place the "Option ===>" or "Command ===>" line near the bottom or top of screen**
  - ✗ Note that certain applications may override this behavior
  - ✗ Also, selecting Command line at bottom will cause long messages to appear on the line above the Command line
- ♦ **<u>Panel display in CUA mode</u> - affects location and display of long message line, command line, and function keys**
- ♦ **<u>Long message in pop-up</u> - display long messages in a pop-up window instead of in the long message line**

Possible Panel Layouts

Command line at bottom

Panel display CUA

Action bar
separator line
title / short message line
data lines
.
.
.
long message line
command/option line
[F key lines]

Panel not display CUA

Action bar
separator line
title / short message line
data lines
.
.
.
[PF key lines]
long message line
command/option line

Command line not at bottom

Panel display CUA

Action bar
separator line
title / short message line
command/option line
long message line
data lines
.
.
.
[F key lines]

Panel not display CUA

Action bar
separator line
title / short message line
command / option line
long message line
data lines
.
.
.
[PF key lines]

- ♦ **If Long message in pop-up is selected, the long message line will not be used to hold long messages, but long messages will appear in a pop-up window**

Additional Settings

❒ **Two tabbing related settings:**

- ♦ **<u>Tab to action bar choices</u> - should the Tab key get you to the action bar**

  - ✗ Note that deselecting this choice allows the Home key to send the cursor to the first input field on a panel instead of to the action bar

  - ✗ This choice is ignored when running in GUI mode

- ♦ **<u>Tab to point-and-shoot fields</u> - let the Tab key take you to fields that are designated as point-and-shoot**

  - ✗ As opposed to just being able to use the arrow keys

  - ✗ When the cursor is at a field designated as point-and-shoot, pressing Enter initiates the action described by the point-and-shoot text

  - ✗ Point-and-shoot fields are designated by a combination of color, intensity, and highlight (user-changeable)

  - ✗ When running in GUI mode, point-and-shoot fields show up as push buttons (command buttons); the Tab key cannot be used to get to point-and-shoot fields in GUI mode

Still More Settings

❒ **Three settings we'll ignore**

- ♦ **Restore TEST/TRACE options - useful when writing your own dialogs**
- ♦ **Session manager mode - not frequently used**
- ♦ **Edit PRINTDS command - way beyond scope of this course**

❒ **One setting we'll refer to later**

- ♦ **Jump from leader dots - discussed when we discuss the Jump function**

❒ **General settings**

- ♦ **Input field pad - designate initial value in input fields; specify B for blanks, N for nulls, or any non-alphanumeric value**
- ♦ **Command delimiter - what character shall be used to delimit commands when you stack multiple commands on a line**
  - ✗ Default is semi-colon (;)
  - ✗ May not specify alphanumeric character (a-z, 0-9), or period (.), or equals sign (=); may not be same as input field pad
  - ✗ Details later when command stacking is discussed
- ♦ **Always show split line - if selected, line shows in split screen mode; if not selected, no split line is displayed**

A Few More Settings

❒ **Some additional, fairly recent additions to the settings list (may have to scroll down do see; you can enter "down" on the command line and press <Enter>)) ...**

- **Enable EURO sign - indicate your terminal supports the Euro currency symbol**

The four options below were introduced in z/OS 1.6

- **Scroll member list - if selected, after you have processed a member of a library from a member list, the list will be scrolled so that member is at the top of the display**

- **Allow empty member list - if not selected, if a library has no members you get a message if you try to look at the member list; if selected, you will see an empty member list**

- **Allow empty member list (nomatch) - same as above in the case where you request a list of members based on matching a pattern instead of just all members**

- **Empty member list for edit only - if not selected, above behavior applies to View, Browse, and other non-edit ways to get member lists**

But Wait - There's More

❐ **In addition to the specific settings you can choose on the Settings panel, the panel has menu choices for further options**

Log/List

- ♦ **Relating to the Log and List data sets - discussed next**

Function keys

- ♦ **Alternatives for display of function key values**

Colors

- ♦ **Allows user to set colors to display**

Environ

- ♦ **Used for tracing and dumping ISPF - not discussed in this class**

Workstation

- ♦ **Used to start and modify GUI sessions**

Identifier

- ♦ **Used to request panel names (id's) and message id's be displayed or not**

Help

- ♦ **The standard list of Help choices**

LIST and LOG Data Sets

❒ **ISPF maintains two data sets for you automatically:**

✗ The <u>LIST</u> data set is allocated whenever you request a print generating function (such as printing a library); the results are not immediately printed but are placed into this data set

✗ The <u>LOG</u> data set is allocated the first time you request a service that generates a log message, such as renaming a data set or submitting a job

❒ **The data set names ISPF uses for the list and log data sets are, respectively:**

userid.SPFn.LIST
userid.SPFLOGn.LIST

where 'n' is an integer from 0 through 9

❒ **Multiple versions are allowed since you are allowed to keep these data sets after you logoff, if you wish**

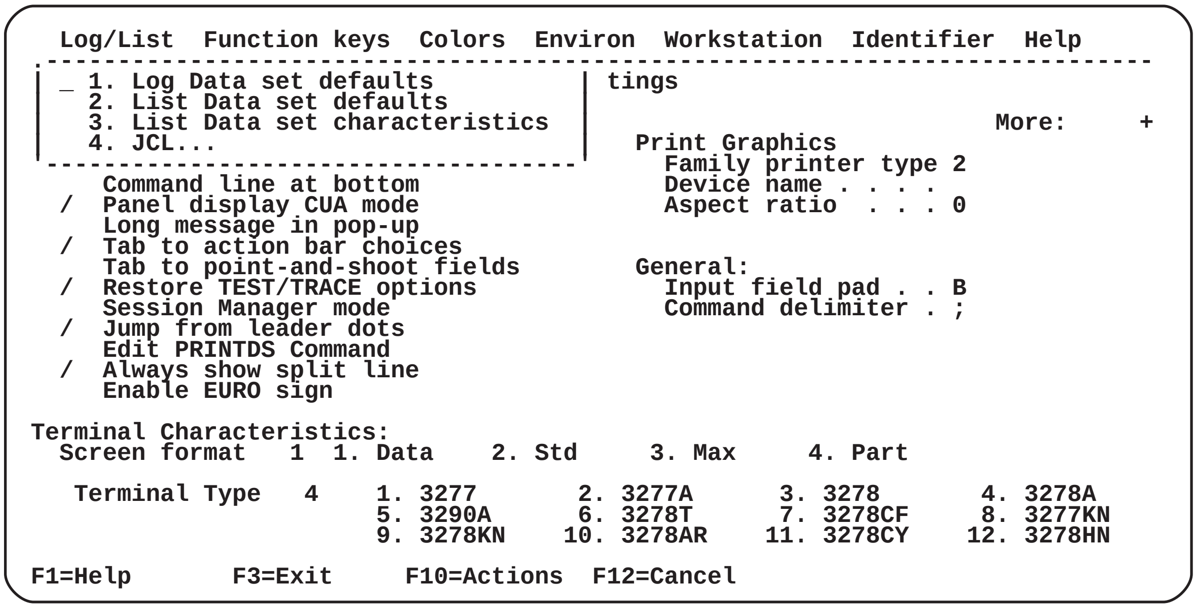
❒ **Alternatively, you may delete or print (then delete) either or both of these data sets at termination time**

- **You can set the default processing for these data sets to meet your own needs, using the Log/List Action bar choice under Settings**

- **If you don't set the default processing, if ISPF created one of these data sets during your session, at logoff time you will be prompted for how you want these data sets to be handled**

LIST and LOG Data Sets, continued

❒ **If you select the Log/List menu option from the Settings action bar, you will see these choices:**



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
 .----------------------------------------------------------------------------
 | _ 1. Log Data set defaults        | tings
 |   2. List Data set defaults       |
 |   3. List Data set characteristics|                            More:     +
 |   4. JCL...                       |   Print Graphics
 '-----------------------------------      Family printer type 2
     Command line at bottom                Device name . . . .
  /  Panel display CUA mode                Aspect ratio  . . . 0
     Long message in pop-up
  /  Tab to action bar choices
     Tab to point-and-shoot fields       General:
  /  Restore TEST/TRACE options            Input field pad . . B
     Session Manager mode                  Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics:
  Screen format   1  1. Data    2. Std     3. Max     4. Part

   Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                        5. 3290A      6. 3278T      7. 3278CF     8. 3277KN
                        9. 3278KN    10. 3278AR    11. 3278CY    12. 3278HN

F1=Help       F3=Exit     F10=Actions  F12=Cancel
```

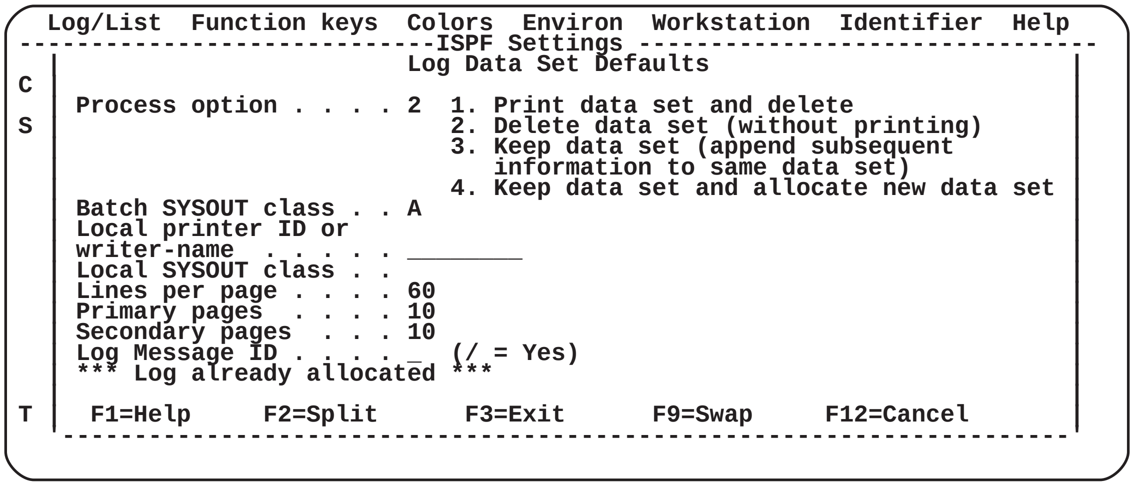
❒ **From here, you can go on to set the Log data set defaults or the List data set defaults**

♦ **You can also set the record length and format of the List data set, and specify a JOB statement used for printing the List or Log data sets when you select the Print option**

✗ We'll not be looking at these last two in this course

LIST and LOG Data Sets, continued

❐ **Menu choice '1', Log Data set defaults, from the Log/List menu, gives you this result:**

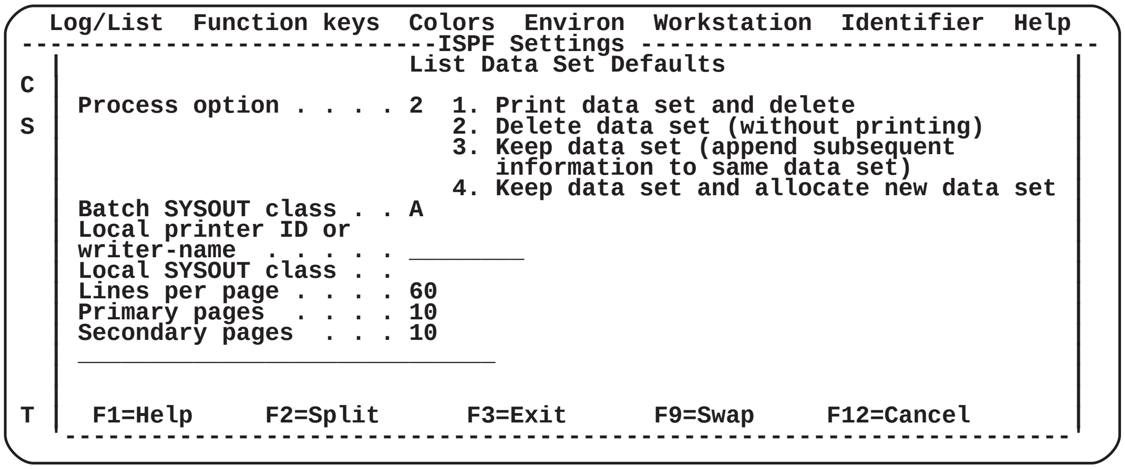


```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
-----------------------------ISPF Settings -------------------------------
  |                       Log Data Set Defaults                          |
C |                                                                      |
  | Process option . . . . 2  1. Print data set and delete               |
S |                           2. Delete data set (without printing)      |
  |                           3. Keep data set (append subsequent        |
  |                              information to same data set)           |
  |                           4. Keep data set and allocate new data set |
  | Batch SYSOUT class . . A                                             |
  | Local printer ID or                                                  |
  | writer-name  . . . . . ________                                      |
  | Local SYSOUT class . .                                               |
  | Lines per page . . . . 60                                            |
  | Primary pages  . . . . 10                                            |
  | Secondary pages  . . . 10                                            |
  | Log Message ID . . . . _  (/ = Yes)                                  |
  | *** Log already allocated ***                                        |
  |                                                                      |
T |  F1=Help     F2=Split      F3=Exit      F9=Swap     F12=Cancel       |
  |----------------------------------------------------------------------|
```

❐ **Fill in the values as necessary**

- ♦ **ISPF will remember these values across logons, until you change them**

❐ **Menu option '2' gives you a very similar screen:**



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
-----------------------------ISPF Settings -------------------------------
  |                       List Data Set Defaults                         |
C |                                                                      |
  | Process option . . . . 2  1. Print data set and delete               |
S |                           2. Delete data set (without printing)      |
  |                           3. Keep data set (append subsequent        |
  |                              information to same data set)           |
  |                           4. Keep data set and allocate new data set |
  | Batch SYSOUT class . . A                                             |
  | Local printer ID or                                                  |
  | writer-name  . . . . . ________                                      |
  | Local SYSOUT class . .                                               |
  | Lines per page . . . . 60                                            |
  | Primary pages  . . . . 10                                            |
  | Secondary pages  . . . 10                                            |
  | _____________________________                                        |
  |                                                                      |
T |  F1=Help     F2=Split      F3=Exit      F9=Swap     F12=Cancel       |
  |----------------------------------------------------------------------|
```

Function Keys Settings

- ❒ **ISPF supports <u>function keys</u> - special keys that can be assigned to an ISPF command string**

    - ♦ **Few PC's have more than 12 function keys, but emulators can often use key combinations (*e.g.*: Shift+F1) to create additional function keys**

- ❒ **ISPF has facilities for changing / setting the commands set to a function key**

    - ♦ **There are also facilities for displaying the settings of function keys on the current panel**

    - ♦ **ISPF also has the ability to package groups of function key assignments (<u>keylists</u>) under a name and to change the current keylist dynamically, under control of the application**

Function Key Settings, continued

❐ **A function key has three attributes associated with it**

- ♦ **Definition - what ISPF command string is associated with the key**
- ♦ **Format - for keylist displays, does this key show on only the long (detailed) function key display or on both long and short displays**
- ♦ **Label - what text should show for the key when the key is displayed (default: first eight characters of Definition)**

❐ **At any point in time, you are operating with the KEYLIST attribute set to ON (keylists are used to determine function key settings) or to OFF (the global PF key setting is used to determine function key settings)**

- ♦ **Using keylists, as you go from panel to panel, the collection of key assignments can change, to reflect the current panel**
  - ✗ IBM supplies eight keylists with ISPF; you cannot delete these, but you can modify them
  - ✗ Application programmers can create, modify, delete their own keylists, when using Dialog Manager (beyond the scope of this course)
- ♦ **Using global PF key settings, there is one set of function key settings in effect as you go from panel to panel**

Function Key Settings, continued

❒ **ISPF has a number of commands associated with function keys, among them:**

KEYLIST {ON | OFF}

♦ **Make keylists active (ON) or inactive (OFF) use the global PF key settings)**

✗ In ISPF 4.2 or later, this command only applies to the application id ("Appl id") currently running; in earlier releases the command is applied to all applications

ZKEYS

♦ **Show a panel for specifying global PF key settings**

KEYS

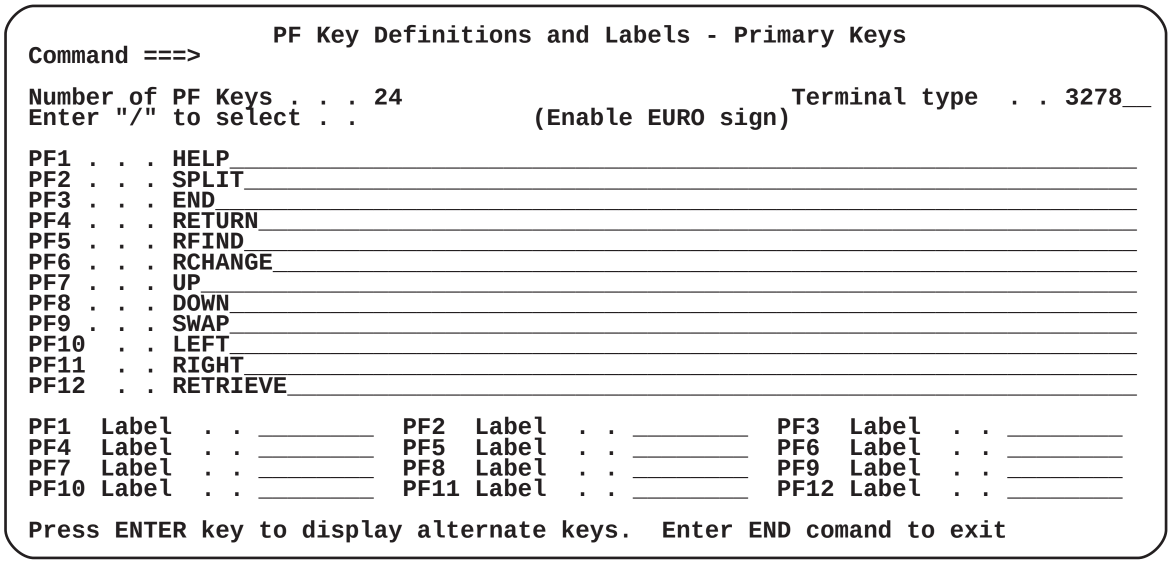
♦ **If keylists are off, same as ZKEYS**

♦ **If keylists are on, show a panel for working with keylists**

❒ **Let's look at the panel for setting global PF keys settings first ...**

Function Key Settings, continued

❒ **The panel for setting PF keys globally looks something like this:**



```
                 PF Key Definitions and Labels - Primary Keys
Command ===>

Number of PF Keys . . . 24                           Terminal type  . . 3278__
Enter "/" to select . .            (Enable EURO sign)

PF1 . . . HELP______________________________________________________________
PF2 . . . SPLIT_____________________________________________________________
PF3 . . . END_______________________________________________________________
PF4 . . . RETURN____________________________________________________________
PF5 . . . RFIND_____________________________________________________________
PF6 . . . RCHANGE___________________________________________________________
PF7 . . . UP________________________________________________________________
PF8 . . . DOWN______________________________________________________________
PF9 . . . SWAP______________________________________________________________
PF10  . . LEFT______________________________________________________________
PF11  . . RIGHT_____________________________________________________________
PF12  . . RETRIEVE__________________________________________________________

PF1  Label  . . ________  PF2  Label  . . ________  PF3  Label  . . ________
PF4  Label  . . ________  PF5  Label  . . ________  PF6  Label  . . ________
PF7  Label  . . ________  PF8  Label  . . ________  PF9  Label  . . ________
PF10 Label  . . ________  PF11 Label  . . ________  PF12 Label  . . ________

Press ENTER key to display alternate keys.  Enter END comand to exit
```

❒ **Note that if your terminal has 24 PF keys, you may first see a panel like the above panel for PF keys 13-24, depending on what keys you designate as primary**

- ♦ **Pressing <Enter> will toggle between the panel for setting primary keys and the panel for setting alternate keys**

❒ **If the label field is left blank, the first eight characters assigned to the PF key will be displayed**

Notes on PF Key Labels

❐ **If the label is specified as 'BLANK', the PF key and equals sign will display, but the value will show as blanks; for example, if**

LABEL2 ===> BLANK

Then...

```
F1=HELP    F2=        F3=END         F4=RETURN    F5=RFIND
```

❐ **If the label is specified as 'NOSHOW', the PF key label will not display at all (the PF key will be omitted from the list at the bottom of the screen); for example, if**

LABEL2 ===> NOSHOW

Then...

```
F1=HELP    F3=END         F4=RETURN    F5=RFIND
```

❐ **If the label has any other value, that is the value that will display for that key; for example, if**

LABEL3 ===> QUIT

Then...

```
F1=HELP    F2=SPLIT       F3=QUIT        F4=RETURN
F5=RFIND
```

PF Key Settings

❒ **Another command related to function keys is PFSHOW**

Syntax

```
PFSHOW   [ON | OFF | TAILOR]
```

Where

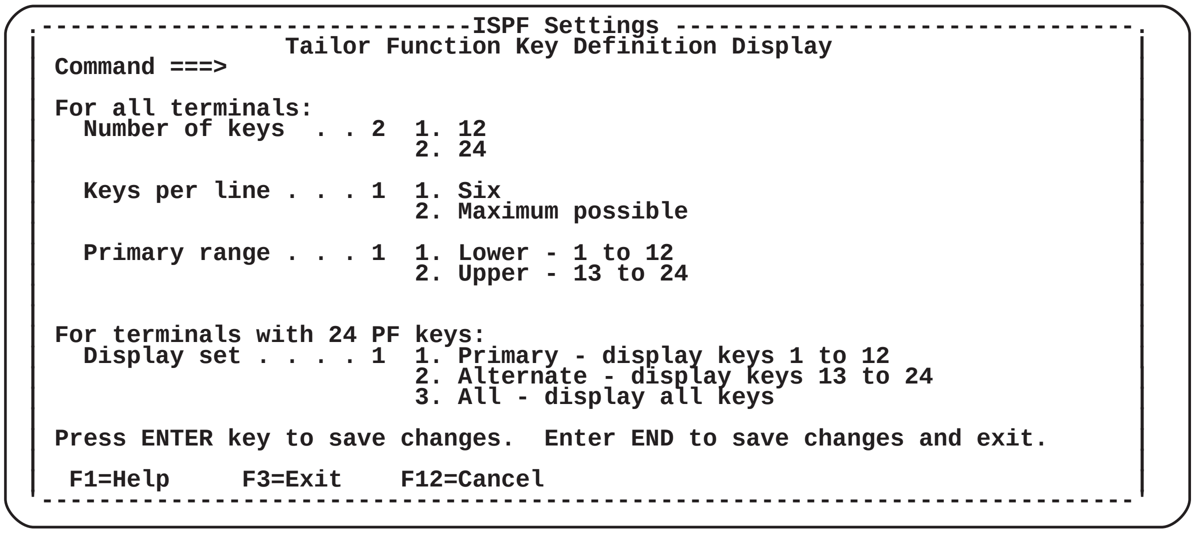
- ♦ **ON - displays, by default, the long form of the function key display**
- ♦ **OFF - removes the function key display from the terminal**
- ♦ **TAILOR - displays a screen where you can specify details of how you want function keys displayed (discussed on the next page)**

Note

- ♦ **If you enter PFSHOW with no operands, the function key display cycles through: long form - short form - off**

PFSHOW TAILOR

❐ **If you enter the PFSHOW TAILOR command, you'll see a screen that looks like this:**



```
.------------------------------ISPF Settings --------------------------------.
|                  Tailor Function Key Definition Display                    |
| Command ===>                                                               |
|                                                                            |
| For all terminals:                                                         |
|   Number of keys  . . 2  1. 12                                             |
|                          2. 24                                             |
|                                                                            |
|   Keys per line . . . 1  1. Six                                            |
|                          2. Maximum possible                               |
|                                                                            |
|   Primary range . . . 1  1. Lower - 1 to 12                                |
|                          2. Upper - 13 to 24                               |
|                                                                            |
|                                                                            |
| For terminals with 24 PF keys:                                             |
|   Display set . . . . 1  1. Primary - display keys 1 to 12                 |
|                          2. Alternate - display keys 13 to 24              |
|                          3. All - display all keys                         |
|                                                                            |
| Press ENTER key to save changes.  Enter END to save changes and exit.      |
|                                                                            |
|  F1=Help     F3=Exit    F12=Cancel                                         |
'----------------------------------------------------------------------------'
```

❐ **Usage of this panel should be pretty self-evident**

FKA

❒ **The last of the commands related to function key display is FKA**

Syntax

FKA [ON | SHORT | OFF | PREFIX | NOPREFIX]

Where

- **ON - displays the long form of the function key display**
- **SHORT - displays the short form of the function key display**
- **OFF - removes function key display**
- **PREFIX / NOPREFIX - apply only when you are running in GUI mode: PREFIX says the push buttons that correspond to function keys should include the "Fn=" text**

  ✗ For example: **F1=Help** versus just **Help**

Note

- **If you enter FKA with no operands, the function key display cycles through: long form - short form - off**

Function Keys Commands - Summary

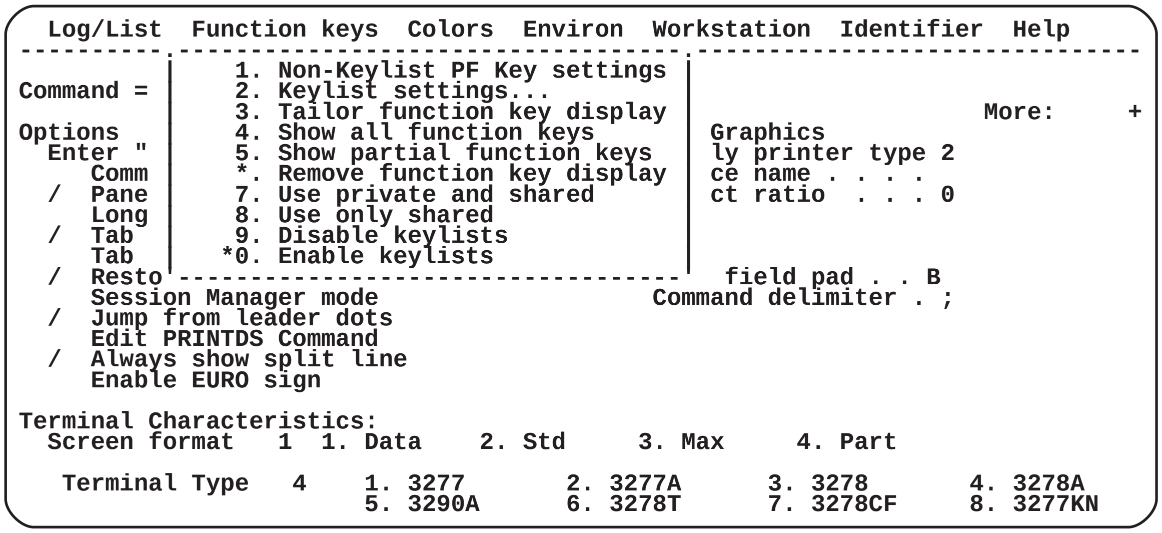
❒ **Just to have them all in one place, here are the keys-related commands we've discussed**

- ♦ **KEYLIST {ON | OFF}**
- ♦ **ZKEYS**
- ♦ **KEYS**
- ♦ **PFSHOW {ON | OFF | TAILOR}**
- ♦ **FKA {ON | SHORT | OFF | PREFIX | NOPREFIX}**

❒ **Now we can relate these to our Settings panel, which is where we left off**

Function Keys and Settings

❒ **From the Settings panel, if you select the Function keys menu item, you'll see these menu choices:**



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
---------.-----------------------------------.-------------------------------
         |     1. Non-Keylist PF Key settings |
Command =|     2. Keylist settings...         |
         |     3. Tailor function key display |                      More:     +
Options  |     4. Show all function keys      | Graphics
  Enter "|     5. Show partial function keys  | ly printer type 2
     Comm|     *. Remove function key display | ce name . . . .
  /  Pane|     7. Use private and shared      | ct ratio  . . . 0
     Long|     8. Use only shared             |
  /  Tab |     9. Disable keylists            |
     Tab |    *0. Enable keylists             |
  /  Resto'-----------------------------------'  field pad . . B
     Session Manager mode                     Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics:
  Screen format   1  1. Data    2. Std     3. Max     4. Part

   Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                        5. 3290A      6. 3278T      7. 3278CF     8. 3277KN
```

Notes

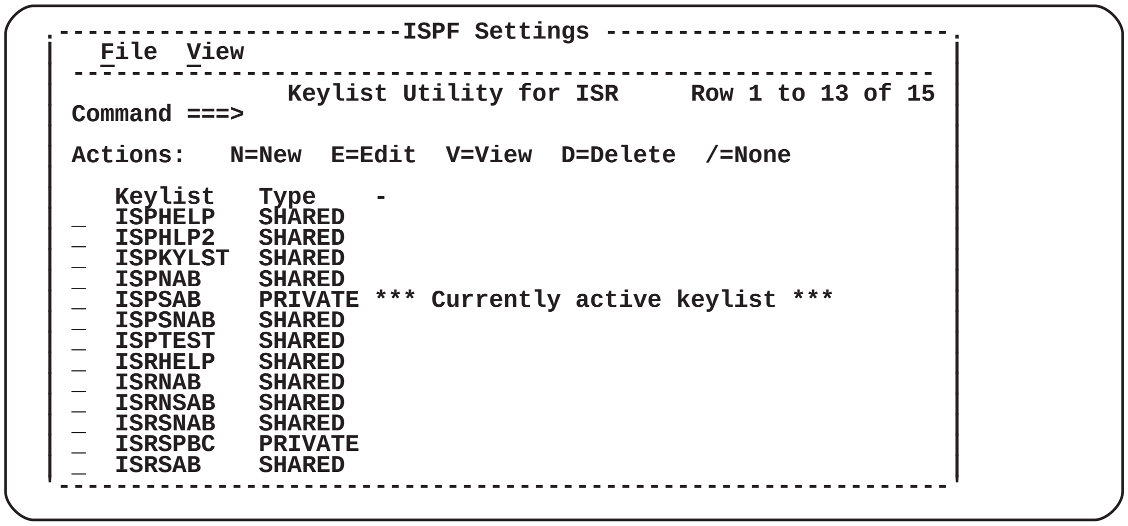
- ♦ **Recall that an asterisk (*) next to a choice indicates the choice is unavailable - in this case because the asterisked options are already in effect**
- ♦ **Option 1 is the same as issuing the ZKEYS command**
- ♦ **Option 2 is discussed on the next page**
- ♦ **Option 3 is the same as issuing PFSHOW TAILOR**
- ♦ **Option 4 is the same as issuing PFSHOW LONG, or FKA LONG, or PFSHOW or FKA with no operands**
- ♦ **Option 5 is the same as issuing FKA SHORT, or FKA or PFSHOW twice, with no operands**
- ♦ **Option 6 is the same as PFSHOW OFF or FKA OFF**
- ♦ **Options 7 and 8 are equivalent to KEYLIST PRIVATE and KEYLIST SHARED, respectively; not discussed further**
- ♦ **Option 9 is the same as KEYLIST OFF**
- ♦ **Option 10 is the same as KEYLIST ON**

Working With Keylists

❐ In addition to turning keylists on and off, and affecting the display of function keys, you can invoke the KEYLIST utility

- ♦ Invoke the KEYLIST command with no operand
- ♦ Select option 2 from the Function keys menu (Keylist settings...)

❐ You'll see a panel like this:



```
.-------------------------ISPF Settings ------------------------.
|  File  View                                                   |
| --------------------------------------------------------------|
|                Keylist Utility for ISR     Row 1 to 13 of 15  |
| Command ===>                                                  |
|                                                               |
| Actions:   N=New  E=Edit  V=View  D=Delete  /=None            |
|                                                               |
|    Keylist   Type    -                                        |
| _  ISPHELP   SHARED                                           |
| _  ISPHLP2   SHARED                                           |
| _  ISPKYLST  SHARED                                           |
| _  ISPNAB    SHARED                                           |
| _  ISPSAB    PRIVATE *** Currently active keylist ***         |
| _  ISPSNAB   SHARED                                           |
| _  ISPTEST   SHARED                                           |
| _  ISRHELP   SHARED                                           |
| _  ISRNAB    SHARED                                           |
| _  ISRNSAB   SHARED                                           |
| _  ISRSNAB   SHARED                                           |
| _  ISRSPBC   PRIVATE                                          |
| _  ISRSAB    SHARED                                           |
'---------------------------------------------------------------'
```

Notes

- ♦ Notice the message "Row 1 to 13 of 15": this panel contains a scrollable list, so you can scroll the list up and down
- ♦ Generally, you tab (or 'arrow') to a key list and key in the appropriate action code
- ♦ The View choice allows you to see other lists of keylists to choose from (not explored here)

Working With Keylists, continued

❐ **The 'Type' column in the list of keylists indicates if a keylist is Shared or Private**

- ♦ **The keylists supplied by IBM are Shared, as are most keylists created by application developers**

- ♦ **If a keylist is Shared, it's definition is kept in a common table of keylists and shared by multiple users**

- ♦ **A Shared keylist cannot be deleted**

- ♦ **If you modify a Shared keylist, a Private copy is made and placed in your own profile pool**

    - ✗ Whenever you work with that keylist, your Private copy will be used instead of the Shared copy

❐ **If you specify a keylist and select the File action bar choice you will see a pull-down menu with these options:**

- ♦ **<u>New</u> - create a whole new keylist (you are shown a panel to code your definition)**
- ♦ **<u>Edit</u> - modify an existing keylist**
- ♦ **<u>View</u> - view a keylist but don't modify it**
- ♦ **<u>Delete</u> - remove an existing keylist (must be Private)**
- ♦ **<u>Exit</u> - leave the keylist utility**

Creating and Changing Keylists

❒ **Creating and changing keylists work pretty much the same**

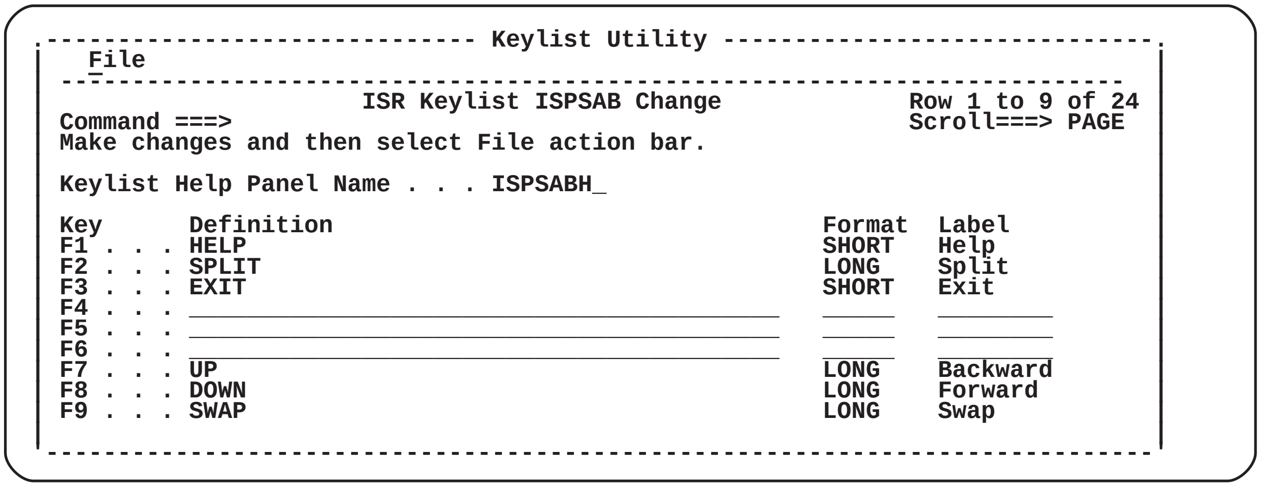
- **When you are changing a keylist, the panel title says "Change", the currently existing values are filled in, and you have an action bar choice of "File"**

  - ✗ **File** gives you two choices: Cancel (forget the changes) and Save and Exit (do what it says)

- **When you are creating a keylist, the panel title says "Create", there are no values specified, and you have action bar choices of "File" and "Defaults"**

  - ✗ **Defaults** lets you initialize your new keylist with certain key settings to begin with

❒ **We use the Change panel for our example:**



```
.------------------------------ Keylist Utility ------------------------------.
|  File                                                                       |
| ------------------------------------------------------------------------    |
|                     ISR Keylist ISPSAB Change            Row 1 to 9 of 24   |
| Command ===>                                             Scroll===> PAGE    |
| Make changes and then select File action bar.                               |
|                                                                             |
| Keylist Help Panel Name . . . ISPSAB_                                       |
|                                                                             |
| Key      Definition                                 Format  Label           |
| F1 . . . HELP                                       SHORT   Help            |
| F2 . . . SPLIT                                      LONG    Split           |
| F3 . . . EXIT                                       SHORT   Exit            |
| F4 . . . ________________________________________   _____   ________        |
| F5 . . . ________________________________________   _____   ________        |
| F6 . . . ________________________________________   _____   ________        |
| F7 . . . UP                                         LONG    Backward        |
| F8 . . . DOWN                                       LONG    Forward         |
| F9 . . . SWAP                                       LONG    Swap            |
|                                                                             |
 -----------------------------------------------------------------------------
```

- **Key in the changes you want, then select File, then Save and Exit**

Final Words About Keylists

❒ You can specify a Help panel name when you create or change a keylist

- ♦ This is used to supply help about the keylist

- ♦ If you enter the KEYSHELP command, the specified Help panel is displayed (if no Help panel was specified, you get a message to that effect)

❒ If you request a delete of a keylist, you will see a panel asking you to confirm that you indeed want to delete the keylist

❒ If you name a keylist and select Browse from the File menu, the panel you see is like the Change panel except you cannot make any changes

❒ Clearly you need to know ISPF or other commands to intelligently set key values

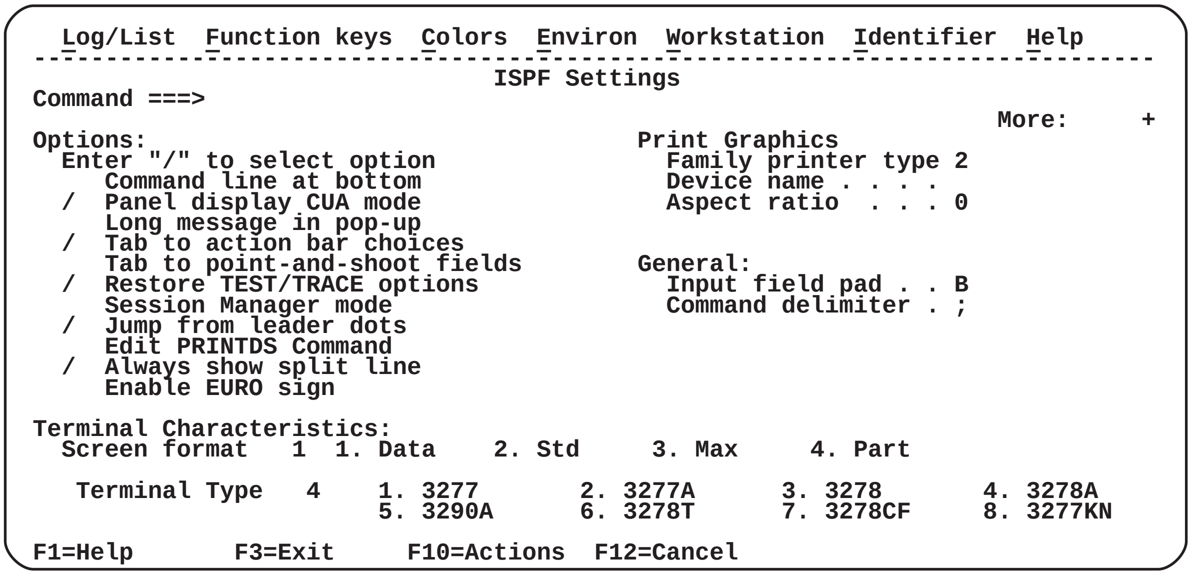
- ♦ We've seen a number of commands so far, and we'll be finding more as we go through the class

Getting Our Bearings

❒ **We've travelled down so many byways in this section that it's probably a good idea to recall what we were doing**

❒ **We began by exploring the ways you can change the look and feel of ISPF to meet your personal preferences**

- ♦ **We have been looking at the Settings panel values**



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
------------------------------------------------------------------------------
                                ISPF Settings
Command ===>
                                                                More:     +
Options:                                  Print Graphics
  Enter "/" to select option                Family printer type 2
     Command line at bottom                 Device name . . . .
  /  Panel display CUA mode                 Aspect ratio  . . . 0
     Long message in pop-up
  /  Tab to action bar choices
     Tab to point-and-shoot fields        General:
  /  Restore TEST/TRACE options             Input field pad . . B
     Session Manager mode                   Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics:
  Screen format   1  1. Data    2. Std     3. Max     4. Part

   Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                        5. 3290A      6. 3278T      7. 3278CF     8. 3277KN

F1=Help       F3=Exit     F10=Actions  F12=Cancel
```

- ♦ **Then we got to exploring the Action bar choices on this panel**

- ♦ **At this point we have examined the Log/List and Function keys choices**

- ♦ **Next we'll examine Colors**

- ♦ **We won't be exploring Environ, since it is for ISPF application developers**

- ♦ **We won't be exploring Workstation, since it requires information specific to your installation and configuration**

- ♦ **The Identifier choice is not significant, and we'll discuss Help elsewhere**

Color, Intensity, and Highlighting

❐ **ISPF uses color, intensity, and highlighting to provide visual clues to the user about the role of various screen components**

❐ **On the host, each component of a panel is assigned an attribute from each of these categories**

- ♦ **Color (white, red, blue, green, pink, yellow, or turquoise)**
- ♦ **Intensity (high, low, or non (that is, not visible))**
- ♦ **Highlighting (none, underscore, reverse, or blinking)**

❐ **For CUA components, ISPF sets default values**

- ♦ **For example, an Action Bar has choices, and a choice may either be selected or not; ISPF assigns color / intensity / highlighting differently to a non-selected Action Bar choice (white / high / none) than to a selected Action Bar choice (yellow / low / none)**

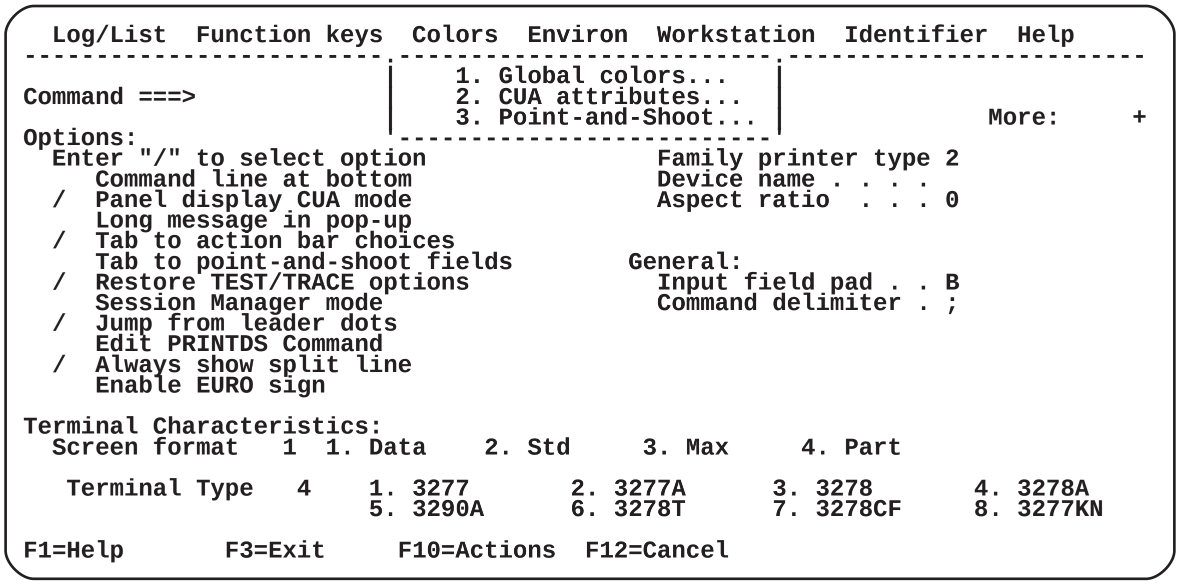
❐ **You can override these settings**

- ♦ **Or you can override color assignments**
- ♦ **Or you can override both CUA settings and color assignments**

❐ **If the terminal being used does not support all the features that ISPF recognizes, ISPF maps the colors, intensity, and highlight values to values the terminal supports**

Colors and Settings

❒ **Going back to the Settings panel, if you choose the Colors action bar choice, you'll see something like this:**



```
  Log/List  Function keys  Colors  Environ  Workstation  Identifier  Help
-------------------------.---------------------------.-------------------------
                         |    1. Global colors...    |
Command ===>             |    2. CUA attributes...   |
                         |    3. Point-and-Shoot...  |              More:     +
Options:                  -------------------------
  Enter "/" to select option                  Family printer type 2
     Command line at bottom                   Device name . . . .
  /  Panel display CUA mode                   Aspect ratio  . . . 0
     Long message in pop-up
  /  Tab to action bar choices
     Tab to point-and-shoot fields          General:
  /  Restore TEST/TRACE options               Input field pad . . B
     Session Manager mode                     Command delimiter . ;
  /  Jump from leader dots
     Edit PRINTDS Command
  /  Always show split line
     Enable EURO sign

Terminal Characteristics:
  Screen format   1  1. Data    2. Std     3. Max     4. Part

   Terminal Type   4    1. 3277       2. 3277A      3. 3278       4. 3278A
                        5. 3290A      6. 3278T      7. 3278CF     8. 3277KN

F1=Help       F3=Exit     F10=Actions  F12=Cancel
```

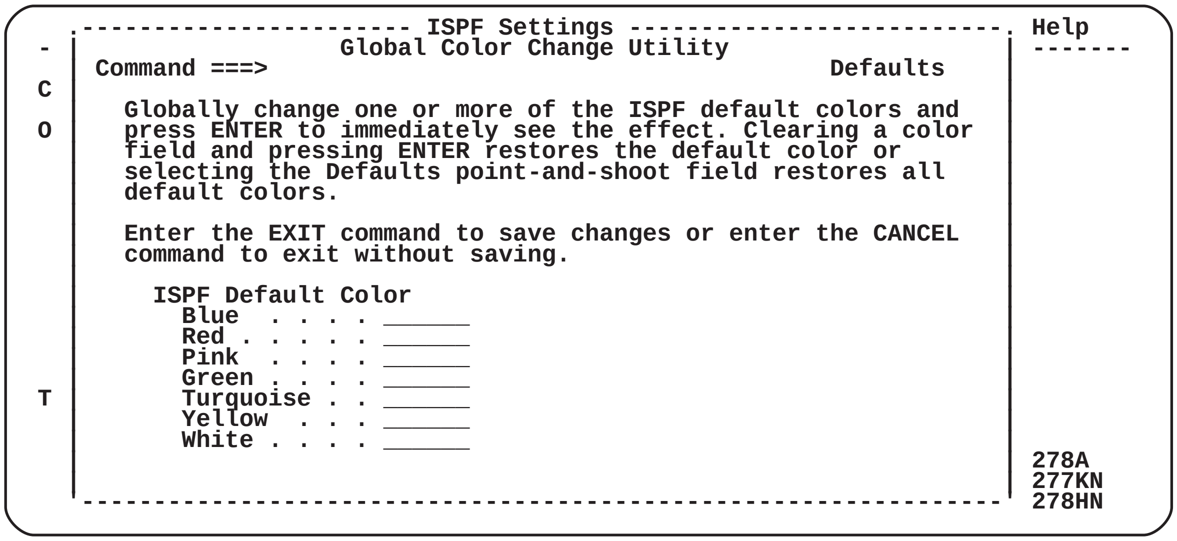
❒ **Choosing Global colors will bring up the Global Color Change Utility panel on the next page (also, the COLOR command will bring up this panel)**

❒ **Choosing CUA attributes will bring up the CUA Attribute Change Utility panel shown two pages ahead (also, the CUAATTR command will bring up this panel)**

❒ **Choosing Point-and-Shoot will also bring up the CUA Attribute Change Utility panel, but it will be positioned at the Point-and-shoot entry in the list (also, the PSCOLOR command will bring up this panel)**

Setting Color Attributes

❒ **ISPF assigns default colors to panel elements that are not CUA elements, using the Global Color Change Utility panel:**



```
  .----------------------- ISPF Settings --------------------------. Help
- |                   Global Color Change Utility                  | -------
  | Command ===>                                        Defaults    |
C |                                                                 |
  |   Globally change one or more of the ISPF default colors and   |
O |   press ENTER to immediately see the effect. Clearing a color  |
  |   field and pressing ENTER restores the default color or       |
  |   selecting the Defaults point-and-shoot field restores all    |
  |   default colors.                                              |
  |                                                                 |
  |   Enter the EXIT command to save changes or enter the CANCEL   |
  |   command to exit without saving.                              |
  |                                                                 |
  |     ISPF Default Color                                         |
  |       Blue  . . . . ______                                     |
  |       Red . . . . . ______                                     |
  |       Pink  . . . . ______                                     |
  |       Green . . . . ______                                     |
T |       Turquoise . . ______                                     |
  |       Yellow  . . . ______                                     |
  |       White . . . . ______                                     |
  |                                                                 | 278A
  |                                                                 | 277KN
  '-----------------------------------------------------------------' 278HN
```

❒ **The instructions are clear how to use this**

- **Note that any color reassignments are applied to CUA objects also**

  ✗ For example, if you assigned RED to selected Action Bar choices and then on this panel assigned GREEN to RED then selected Action Bar choices will display as GREEN!

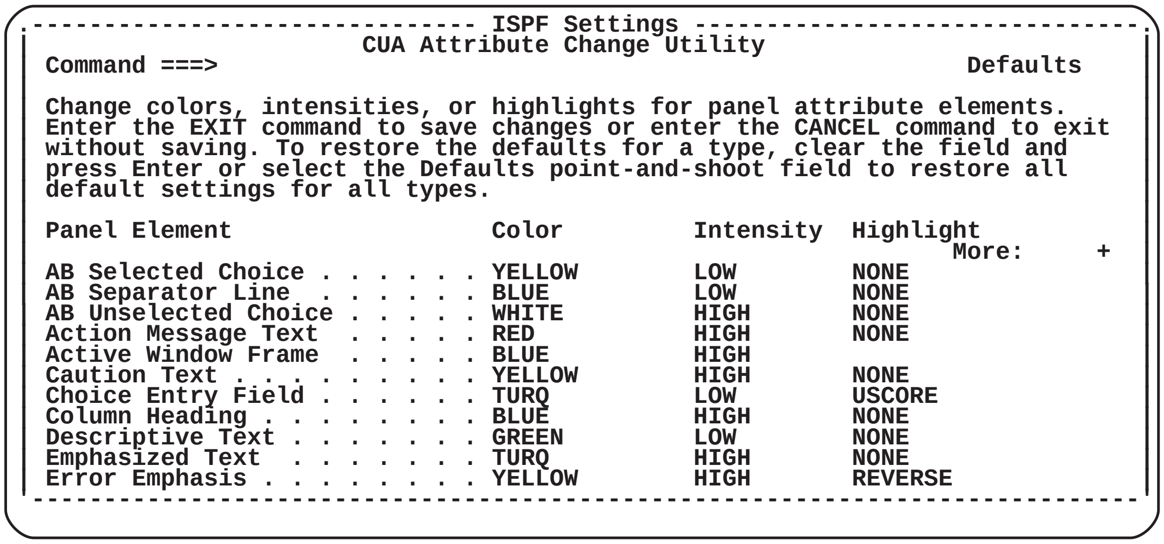
- **Note also that color is managed by ISPF and your emulator (if you are using one), and there may be conflicts in behavior**

- **Notice the word "Defaults" near the upper right corner: this is our first real example of a point-and-shoot field**

  ✗ Move the cursor to that word and press ENTER, and all the panel elements will be reset to the IBM supplied default values

Setting CUA Attributes

❒ **You change CUA Attributes using the CUA Attribute Change Utility Panel:**



```
.------------------------------- ISPF Settings -------------------------------.
|                      CUA Attribute Change Utility                            |
| Command ===>                                                    Defaults     |
|                                                                              |
| Change colors, intensities, or highlights for panel attribute elements.      |
| Enter the EXIT command to save changes or enter the CANCEL command to exit   |
| without saving. To restore the defaults for a type, clear the field and      |
| press Enter or select the Defaults point-and-shoot field to restore all      |
| default settings for all types.                                              |
|                                                                              |
| Panel Element               Color         Intensity  Highlight              |
|                                                          More:     +        |
| AB Selected Choice . . . . . . YELLOW        LOW        NONE                 |
| AB Separator Line  . . . . . . BLUE          LOW        NONE                 |
| AB Unselected Choice . . . . . WHITE         HIGH       NONE                 |
| Action Message Text  . . . . . RED           HIGH       NONE                 |
| Active Window Frame  . . . . . BLUE          HIGH                            |
| Caution Text . . . . . . . . . YELLOW        HIGH       NONE                 |
| Choice Entry Field . . . . . . TURQ          LOW        USCORE               |
| Column Heading . . . . . . . . BLUE          HIGH       NONE                 |
| Descriptive Text . . . . . . . GREEN         LOW        NONE                 |
| Emphasized Text  . . . . . . . TURQ          HIGH       NONE                 |
| Error Emphasis . . . . . . . . YELLOW        HIGH       REVERSE              |
'------------------------------------------------------------------------------'
```

Notes

♦ **Notice below the Highlight column header the word "More" with a plus sign (+) next to it**

✗ This is a CUA way of saying the list is scrollable

✗ The “+” says you can scroll forward; a “-” says you can scroll backward; if you have both: “- +”, you can scroll both ways

✗ We'll talk more about scrollable panels later

Look and Feel: Options

❒ **Recall the Primary Option Menu looks like this:**

```
  Menu  Utilities  Compilers  Options  Status  Help
------------------------------------------------------------------------------
                           ISPF Primary Option Menu               Enter option
Option ===>

0  Settings      Terminal and user parameters            User ID . : STNT329
1  View          Display source data or listings         Time. . . : 12:25
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 : CTP
7  Dialog Test   Perform dialog testing                  TSO prefix: STNT329
9  IBM Products  IBM program development products        System ID : SYUB
10 SCLM          SW Configuration Library Manager        MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell      Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications

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

❒ **In our quest for tailoring the look and feel of ISPF, we have explored the choices from the Settings option**

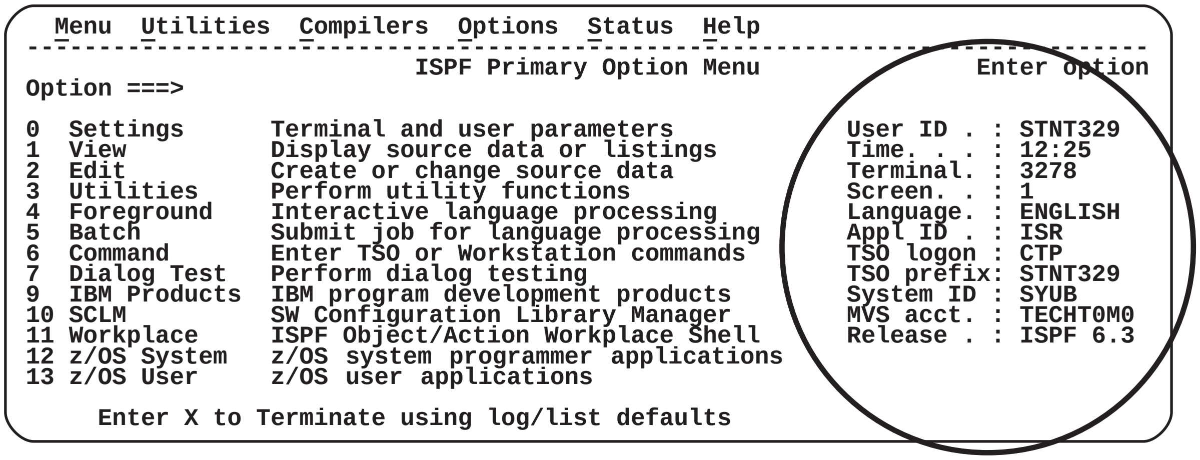
❒ **You can make some of the same choices from the Options action bar menu:**

```
  Menu  Utilities  Compilers  Options  Status  Help
--------------------------- .-----------------------------. -----------------
                            |    1. General Settings      |
Option ===>                 |    2. CUA Attributes...     |
                            |    3. Keylists...           |
0  Settings      Terminal a |    4. Point-and-Shoot...    |  ID . : STNT329
1  View          Display so |    5. Colors...             | . . . : 12:25
2  Edit          Create or  |    6. Dialog Test appl ID...| inal. : 3278
3  Utilities     Perform ut '-----------------------------'  en. . : 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 : CTP
7  Dialog Test   Perform dialog testing                  TSO prefix: STNT329
9  IBM Products  IBM program development products        System ID : SYUB
10 SCLM          SW Configuration Library Manager        MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell      Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications

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

- ♦ **Option 1 invokes the Settings panel**
- ♦ **Option 2 invokes the CUA Attribute Change Utility panel**
- ♦ **Option 3 invokes the Keylist Utility panel**
- ♦ **Option 4 invokes the CUA Attribute Change Utility panel, positioned at the Point-and-shoot entry**
- ♦ **Option 5 invokes the Global Color Change Utility panel**
- ♦ **Option 6 is used by Dialog Manager developers**

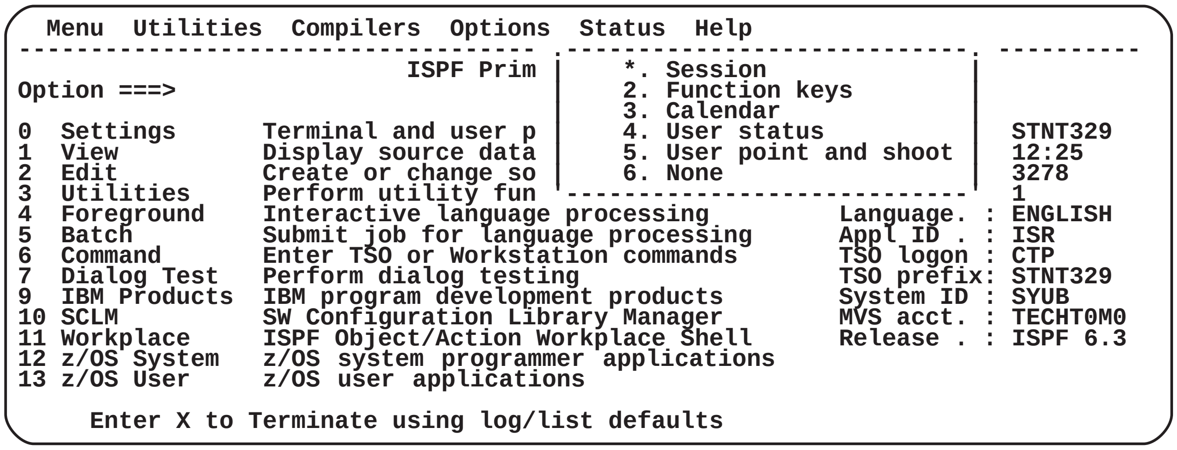
Look and Feel: Status

❒ **Finally(!), we need to point out that the area of the Primary Option Menu circled below is called the Status area:**



```
  Menu  Utilities  Compilers  Options  Status  Help
------------------------------------------------------------------------------
                           ISPF Primary Option Menu                Enter option
Option ===>

0  Settings      Terminal and user parameters          User ID . : STNT329
1  View          Display source data or listings       Time. . . : 12:25
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 : CTP
7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
9  IBM Products  IBM program development products      System ID : SYUB
10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell    Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications

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

❒ **You can tailor the information displayed in the Status area by selecting the Status action bar choice:**



```
  Menu  Utilities  Compilers  Options  Status  Help
------------------------------------ .---------------------------. ----------
                           ISPF Prim |    *. Session             |
Option ===>                          |    2. Function keys       |
                                     |    3. Calendar            |
0  Settings      Terminal and user p |    4. User status         |  STNT329
1  View          Display source data |    5. User point and shoot|  12:25
2  Edit          Create or change so |    6. None                |  3278
3  Utilities     Perform utility fun '---------------------------'  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 : CTP
7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
9  IBM Products  IBM program development products      System ID : SYUB
10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
11 Workplace     ISPF Object/Action Workplace Shell    Release . : ISPF 6.3
12 z/OS System   z/OS system programmer applications
13 z/OS User     z/OS user applications

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

❒ **Remember, the current option will be shown as unselectable; so far we have displayed the Session information in the status area**

- ♦ **But you can request different information to display (options 2 through 5) or no information to display (option 6)**
- ♦ **The details are beyond what we want to discuss, but it's good to be aware of this**

Look and Feel: Summary

❐ We have seen a variety of attributes you can change with the ISPF interface:

- ♦ Command line at bottom (or not)
- ♦ Panel display in CUA mode (or not)
- ♦ Long messages to display in pop-up windows (or not)
- ♦ Tab to action bar (or not)
- ♦ Tab to point-and-shoot fields (or not)
- ♦ Input field pad (your choice)
- ♦ Command delimiter (your choice)
- ♦ Log and List data set handling (variety of options)
- ♦ Function keys (assignments, labels, displays)
- ♦ Colors (global and CUA)
- ♦ Status area (several choices)

❐ And we've seen that we can change the look and feel through several interfaces

- ♦ Menu options
- ♦ Action bar choices
- ♦ Commands

❐ Here's a chance to practice using these facilities...

Computer Exercise: Changing the Look and Feel of ISPF

Get into ISPF and experiment with various alternatives for the look and feel of ISPF behavior. Set the combination of options that work best for you. Some suggestions:

1. Try the command line at the top, and at the bottom; leave it where you prefer

2. We recommend you deselect Tab to action bar choices; this is because with this setting the Home key will not send the cursor to the action bar

3. Try using an input field pad character of a period (.), a backward slash (\), an exclamation point (!), and a plus sign (+); choose a character that you prefer (it might be one of the above, or it might be B (for blank), N (for null), or any other special character except a forward slash (/)

4. Set your Log and List default processing to Delete; this prevents excessive listings printing out in the computer room; when you really want a listing to print you can use the Log and List commands (discussed later)

5. Play with function keys, keylists, and colors.

Exercise stretch: experiment with status area values of Function keys and Calendar to see what they display like.

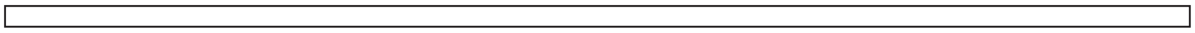
Section Preview

❐ **Working With Data**

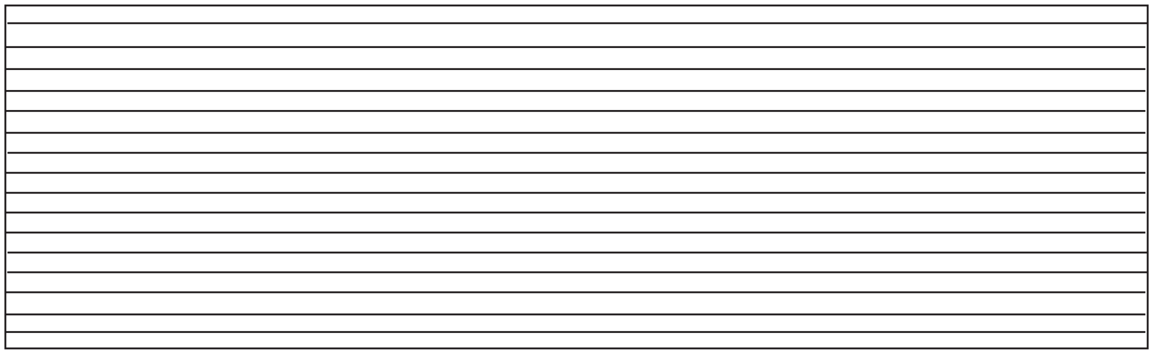
- ♦ **Files and Data Sets**
- ♦ **Data Set Organizations**
- ♦ **Data Set Naming**
- ♦ **Locating Data Sets, [members,] Records**
- ♦ **Data Set Properties**
- ♦ **SMS — Storage Management Subsystem**
- ♦ **Finding a Data Set's Properties**
- ♦ **Finding a Data Set's Properties (Machine Exercise)**

Files and Data Sets

❐ The data that describe a single entity (person, item, customer, supplier, and so on) can be organized as a string of fields, which, when strung together could be pictured as a line, called a record:



❐ A set of related records is called a file (the personnel file, the inventory file, the customer file, and so on), which could be pictured as a list of records, like:



❐ While most of the industry refers to a collection of related records as a file, IBM has traditionally used the term data set

- ♦ For our purposes, the terms can be used as synonyms

Data Set Organizations

❒ **A data set can be written on magnetic tape, disk, or even paper (a report, or print file)**

❒ **For most media, the records in the list (file) must be written one after the other, and read the same way: one after the other**

- **We say the file is organized sequentially; it is a <u>sequential data set</u>**

❒ **For disk and CD-ROM type devices (we also use the term DASD: Direct Access Storage Device), data may be recorded sequentially**

- **But there are many other possibilities**
- **In this course, we focus on sequential data sets and another type of data set organization called <u>partitioned</u>**

Partitioned Data Sets

❒ **A partitioned data set (or, of course, PDS) occupies a chunk of disk space logically divided into a Directory and space for Members**

- ♦ **Members themselves look like sequential files, except for the fact they are stored in the space of the PDS instead of out there on their own**

- ♦ **The directory is a set of data blocks that holds the names of all the members in the PDS, along with a pointer to where the corresponding member's records are found**

  - ✗ In many places in ISPF, the directory of a library is referred to as the index

❒ **Later we'll discuss an enhancement to PDSs called PDSEs (for Partitioned Data Set Extended)**

- ♦ **The term library is sometimes used to mean either a PDS or PDSE**

- ♦ **From a day to day work perspective, PDSs and PDSEs look to be the same**

Data Set Naming

❒ A qualifier is a string of 1-8 alphanumeric or national ($ # @) characters, the first of which is not numeric

❒ A data set name (DSNAME) consists of 1 or more qualifiers, separated by periods, up to a maximum of 44 characters

Examples:

MYFILE

SYS1.LINKLIB

$TRNCM.TRAIN.LIBRARY

DEPT56.PAYROLL.EXTRACT.JANUARY.TEMP#1

❒ We use the term "level" to indicate where in a data set name a qualifier is, and you'll hear expressions like

- High level qualifier (= leftmost qualifier)
- Low level qualifier (= rightmost qualifier)
- Fully-qualified data set name

TSO Data Set Naming Conventions

A Data set name consists of three qualifiers:

PREFIX.FILENAME.TYPE

PREFIX:

Default to current userid

FILENAME:

User choice

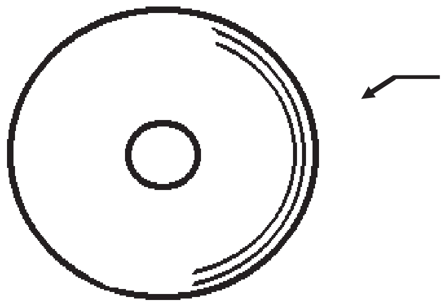
TYPE:

From list of standard choices, or user choice

TSO Standard Types List

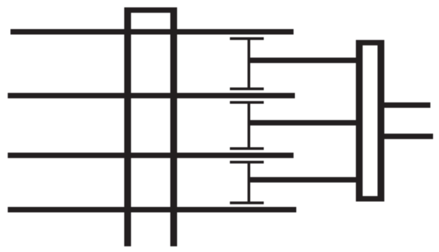
Type	Implied Data Set Contents
ASM	Assembler source
CLIST	TSO commands / CLIST statements
CNTL	JCL and SYSIN for SUBMIT command
COBOL	COBOL source
DATA	Uppercase text
EXEC	REXX exec's
FORT	FORTRAN source
LINKLIST	Output listing from the Linkage Editor
LIST	Listings from assemblies / compiles
LOAD	Load modules
LOADLIST	Output from Loader program
OBJ	Object modules
OUTLIST	Listings from OUTPUT command
PASCAL	PASCAL source
PLI	PL/I source
TESTLIST	Listings from TEST command
TEXT	Uppercase and lowercase text
VSBASIC	BASIC source

DASD Concepts



Platter

Cylinder concept



❒ Direct Access Storage Device

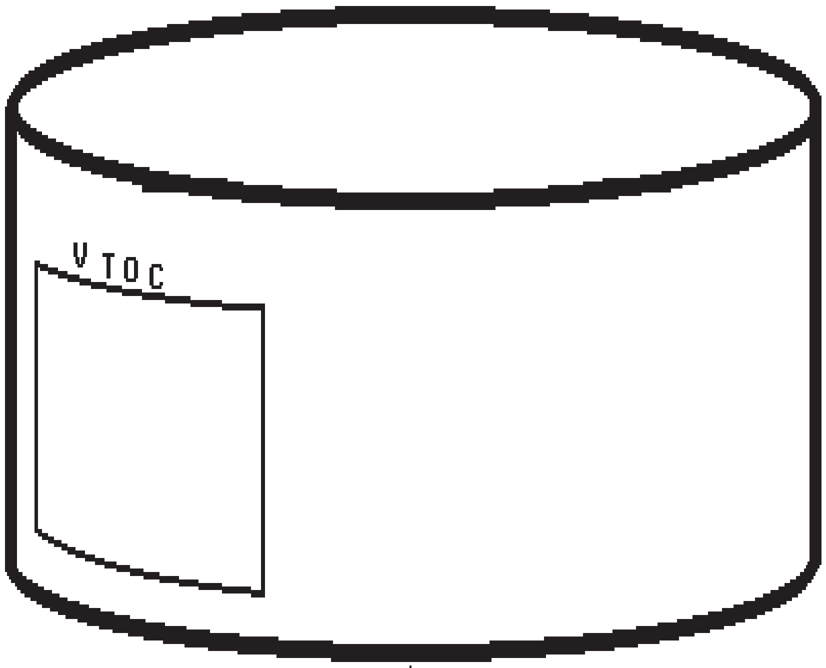
Volume Table Of Contents (VTOC)

❒ **Small data set on every DASD volume**

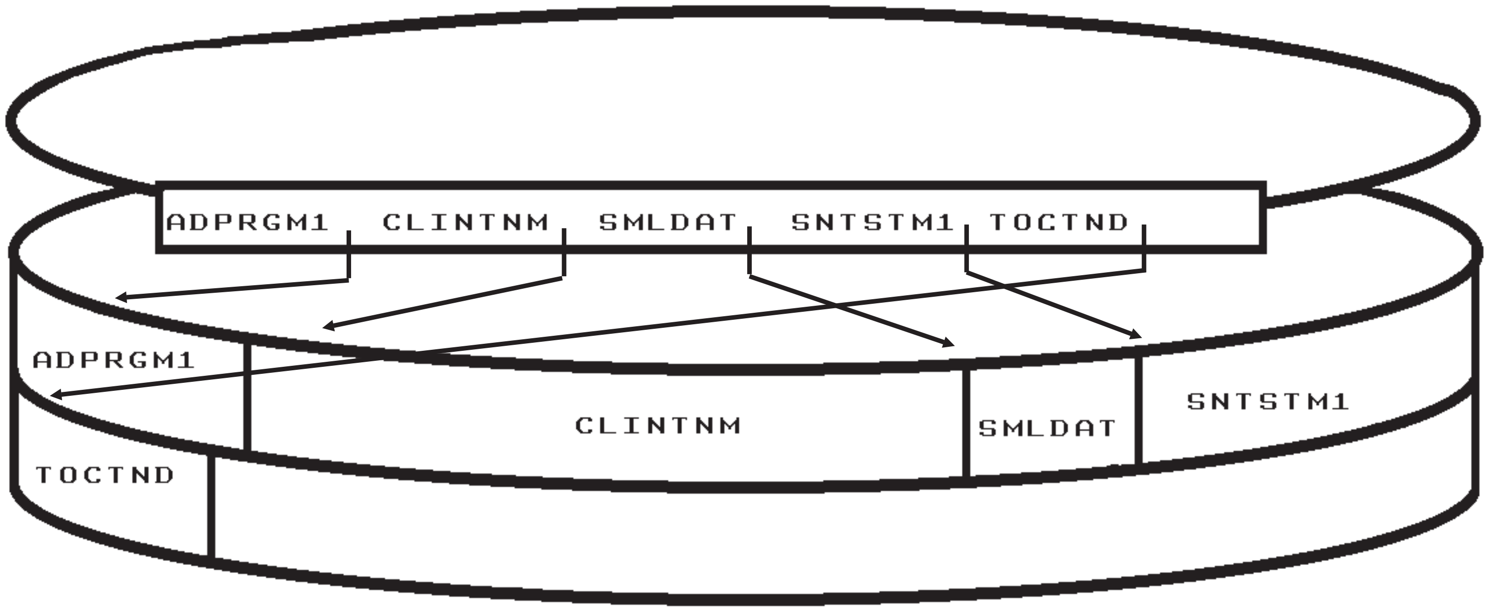
❒ **Contains description of all files on the volume**

❒ **Contains description of all available (free) space on the volume**

❒ **Records are called Data Set Control Blocks (DSCB's)**

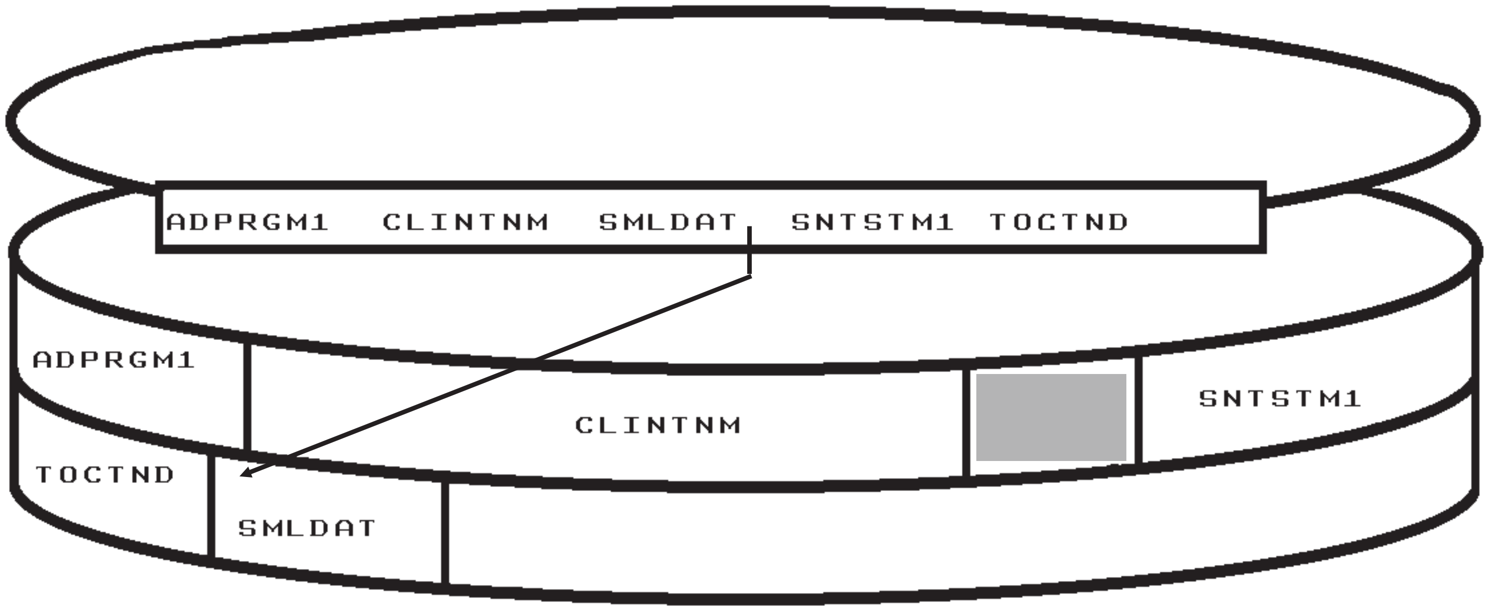


A Partitioned Data Set (PDS)



- ❒ Some (user-chosen) number of directory blocks are placed at the front of the data set
- ❒ Directory blocks contain the list of members, and pointers to where the members reside within the data set

The Effects Of Updating



- ❐ When a member is updated, the new version is written at the end of the data set and the pointer in the directory changed
- ❐ The space previously occupied by the member is unused and unusable until the PDS is compressed

A Catalog

❐ A list of data set names the system is to keep track of

❐ Each data set name is associated with the device type (*e.g.*: 3490, 3380, 3390) and the volume serial(s) that identify where the dataset resides

- ♦ Volume serials are 1-6 characters long
- ♦ Not necessarily numeric, for example:

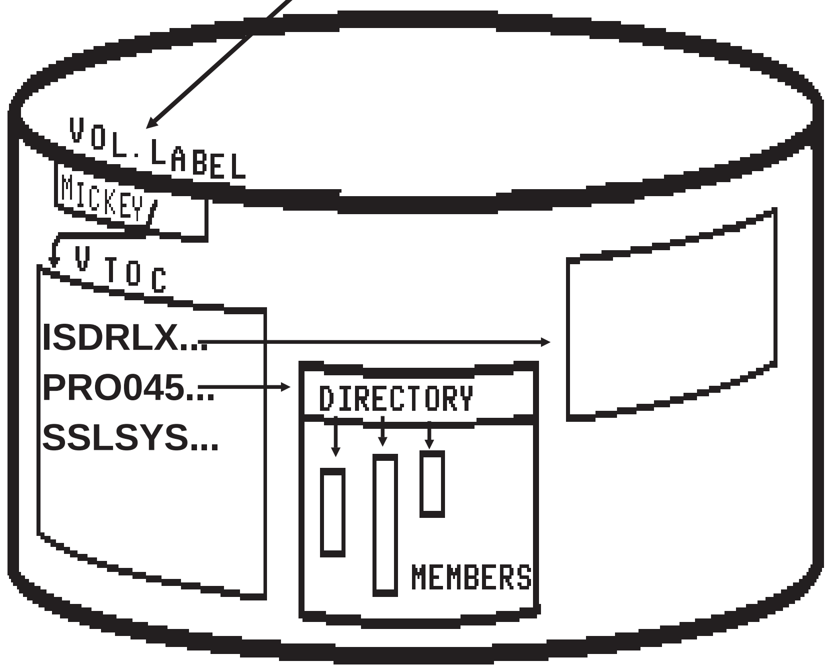
```
SPOOL1
SYSRES
111111
MICKEY
DOPEY
DAFFY
```

❐ Catalogs are themselves kept on disk

Locating Data Sets, [members,] Records

CATALOG

ISDRLX.CUSTFILE.MASTER	DASD	MICKEY
PRO045.CODES.KRGLAGR	DASD	MICKEY
$TRNCM.PGMS.DVLP.LOADLIB.T3	TAPE	SLEEPY
•	•	•
•	•	•
•	•	•



Data Set Properties

❒ **Every data set can be described by a set of <u>characteristics</u>, or <u>properties</u>**

- ♦ **Some of these are of no interest to us, and can be safely ignored**

❒ **But some properties are useful and important to us in our day to day work, and we survey these characteristics now**

- ♦ **Data set name**
  - ✗ What is the name of the data set? This is how the system will refer to the data set
  - ✗ Note that some data sets do not have names (print files for example)
- ♦ **Data set location**
  - ✗ Where is the data set found?
    - ➢ Print data set (also called SYSOUT data)
    - ➢ In-stream data (mixed in with JCL)
    - ➢ Tape (cataloged or non-cataloged)
    - ➢ DASD (always cataloged)
    - ➢ Member of a PDS / PDSE?
    - ➢ For tape or DASD: which volume (Volume Serial)

Data Set Properties, 2

❒ **Additional useful / interesting data set properties ...**

- ♦ **Data set organization ('DSORG' in the literature)**
  - ✗ Sequential (say 'PS' for Physical Sequential)
  - ✗ Library (say 'PO' for Partitioned Organization)
    - ➢ PDS (say DSNTYPE of 'PDS')
    - ➢ PDSE (say DSNTYPE of 'LIBRARY')
  - ✗ Others not discussed in this class (e.g.: VSAM)

- ♦ **Data set size**
  - ✗ How much **space** should be reserved (allocated) for a new data set
    - ➢ For tape, how many volumes (reels or cartridges)
    - ➢ For DASD, can specify in bytes, kilobytes (thousands of bytes), megabytes (millions of bytes), blocks, tracks, or cylinders
    - ➢ Figure: size of logical records, in bytes (LRECL) times number of records in the data set
    - ➢ For a PDS, how many directory blocks should be reserved? (based on how many members are expected in the PDS)
  - ✗ For an existing data set, useful to know how much space is allocated versus how much space is actually being used (for a library, how many members are currently there)

Data Set Properties, 3

❒ **Other useful / interesting properties of data sets ...**

- ♦ **Record format ('RECFM' in the literature)**
    - ✗ Are records all the same size (Fixed length records)
    - ✗ Are records of differing sizes (Variable length records)
    - ✗ Are records in the mysterious Undefined format
    - ✗ For fixed length and variable length records, are the records Blocked or unblocked
    - ✗ For fixed length and variable length records, does the first byte contain an ANSI carriage control character for a printer
    - ✗ Specify a record format through a string of letters, most common: F, FB, FA, FBA, V, VB, VA, VBA, U

- ♦ **Logical record length ('LRECL' in the literature)**
    - ✗ Number of bytes in each record
    - ✗ For variable length records, the largest record size (data bytes) + 4

Data Set Properties, 4

- **Still more useful / interesting properties of data sets ...**

  - **Block size ('BLKSIZE' in the literature)**

    - Number of bytes in a block

    - Large blocks use tape and DASD space more efficiently

    - Usually calculated by system when a file is created, but can be explicitly specified if there is a special requirement (very unusual in modern systems)

  - **Creation date / Expiration date**

    - System automatically records date of creation

    - Expiration date indicates when data set is eligible to be deleted (scratched)

    - Can specify a retention period (RETPD) or an actual expiration date (EXPDT) when file is created

- **But wait - there's more ...**

SMS — Storage Management Subsystem

❒ **Many installations use a product called SMS**

- ♦ **SMS means "Storage Management Subsystem", in some contexts, and "System Managed Storage" in others**
- ♦ **Full name is DFSMS (Data Facility Storage Management Subsystem); we'll simply say SMS**

❒ **SMS helps an organization manage its data by allowing the organization to characterize groups of data sets and then simply assigning new data sets to these various groups**

❒ **We'll fill in more details later, but for now, observe that SMS introduces three additional properties for data sets that are SMS-managed**

- ♦ **Management class ('MGMTCLAS')**
  - ✗ Implies characteristics for migration, retention, deletion, etc.
- ♦ **Data class ('DATACLAS')**
  - ✗ Implies physical characteristics of data set
- ♦ **Storage class ('STORCLAS')**
  - ✗ Implies performance characteristics of data set

Finding a Data Set's Properties

❒ **If you are interested in finding out the properties for an existing data set, you can tell ISPF the name of the data set**

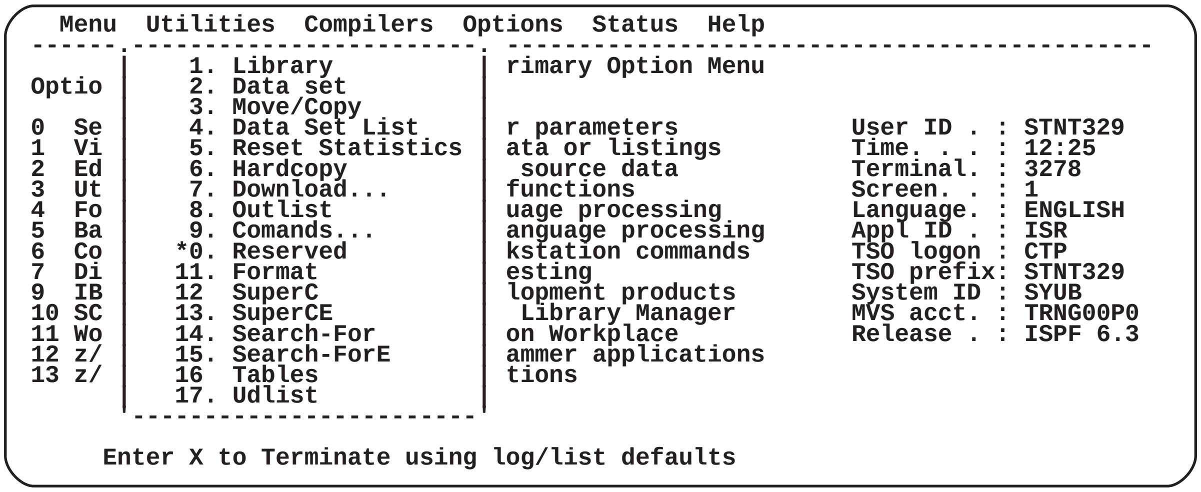
- ♦ **There are a variety of ways for doing this and we'll explore them shortly**

❒ **The system then locates information, as follows**

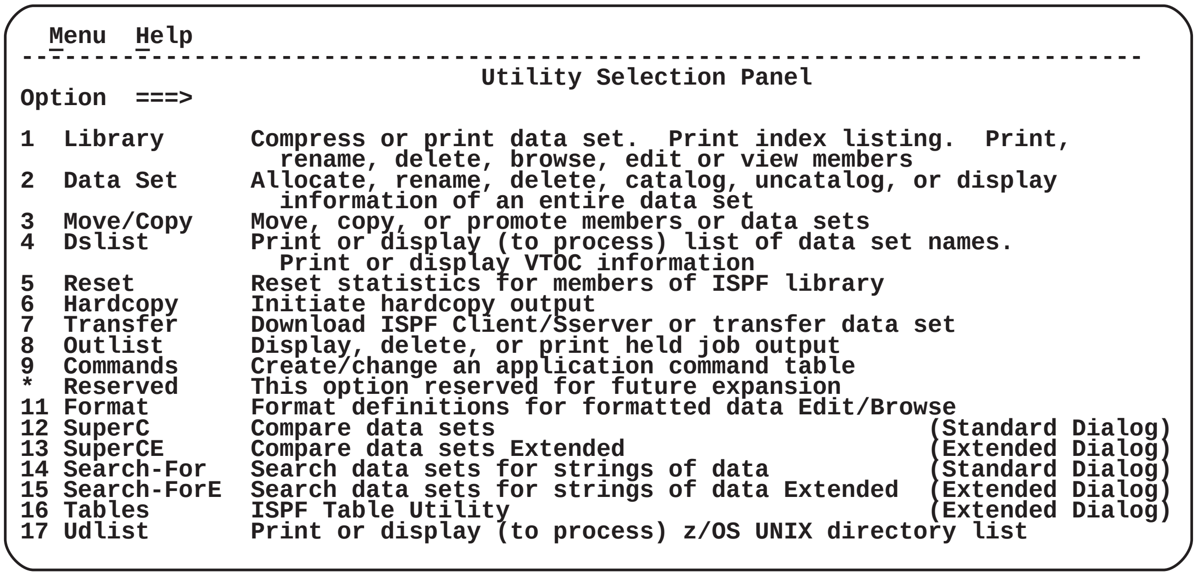
- ♦ **Look up the data set name in the catalog, find the unit type and volume serial**
- ♦ **For DASD, go to the volume, look at the volume label, which points to the VTOC on the volume, which contains labels (DSCBs) for all the files on the volume**
- ♦ **The label contains the physical characteristics (e.g.: LRECL, RECFM, DSORG, space) and location of the data on the volume**
- ♦ **If the data set is a library, the directory is at the front of the library and contains member names and the location for each member in the data set**
  - ✗ The directory might also contain statistics for some or all members (date member created, date last modified, number of lines (records), TSO id of user who last modified the member, and so on)
- ♦ **For tape, the system asks for the tape to be mounted, and the data set label contains the physical characteristics (e.g.: LRECL, RECFM, DSORG)**

Finding a Data Set's Properties, 2

❒ **One way to find out a sequential data set or PDS's properties is to use ISPF Utilities, which you can get to from Option 3 of the Primary Option Menu or from the “Utilities” Action bar pull-down:**


```
  Menu  Utilities  Compilers  Options  Status  Help
------.------------------------. ------------------------------------------
      |     1. Library         | rimary Option Menu
Optio |     2. Data set        |
      |     3. Move/Copy       |
0  Se |     4. Data Set List   | r parameters           User ID . : STNT329
1  Vi |     5. Reset Statistics| ata or listings        Time. . . : 12:25
2  Ed |     6. Hardcopy        |  source data           Terminal. : 3278
3  Ut |     7. Download...     | functions              Screen. . : 1
4  Fo |     8. Outlist         | uage processing        Language. : ENGLISH
5  Ba |     9. Comands...      | anguage processing     Appl ID . : ISR
6  Co |    *0. Reserved        | kstation commands      TSO logon : CTP
7  Di |    11. Format          | esting                 TSO prefix: STNT329
9  IB |    12  SuperC          | lopment products       System ID : SYUB
10 SC |    13. SuperCE         |  Library Manager       MVS acct. : TRNG00P0
11 Wo |    14. Search-For      | on Workplace           Release . : ISPF 6.3
12 z/ |    15. Search-ForE     | ammer applications
13 z/ |    16  Tables          | tions
      |    17. Udlist          |
      '------------------------'

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

❒ **Using the Primary Option Menu choice 3 sends you to the Utilities main menu, which is an expanded version of the Utilities pull-down menu:**


```
  Menu  Help
------------------------------------------------------------------------------
                                Utility Selection Panel
Option  ===>

1  Library      Compress or print data set.  Print index listing.  Print,
                  rename, delete, browse, edit or view members
2  Data Set     Allocate, rename, delete, catalog, uncatalog, or display
                  information of an entire data set
3  Move/Copy    Move, copy, or promote members or data sets
4  Dslist       Print or display (to process) list of data set names.
                  Print or display VTOC information
5  Reset        Reset statistics for members of ISPF library
6  Hardcopy     Initiate hardcopy output
7  Transfer     Download ISPF Client/Sserver or transfer data set
8  Outlist      Display, delete, or print held job output
9  Commands     Create/change an application command table
*  Reserved     This option reserved for future expansion
11 Format       Format definitions for formatted data Edit/Browse
12 SuperC       Compare data sets                             (Standard Dialog)
13 SuperCE      Compare data sets Extended                    (Extended Dialog)
14 Search-For   Search data sets for strings of data          (Standard Dialog)
15 Search-ForE  Search data sets for strings of data Extended  (Extended Dialog)
16 Tables       ISPF Table Utility                            (Extended Dialog)
17 Udlist       Print or display (to process) z/OS UNIX directory list
```

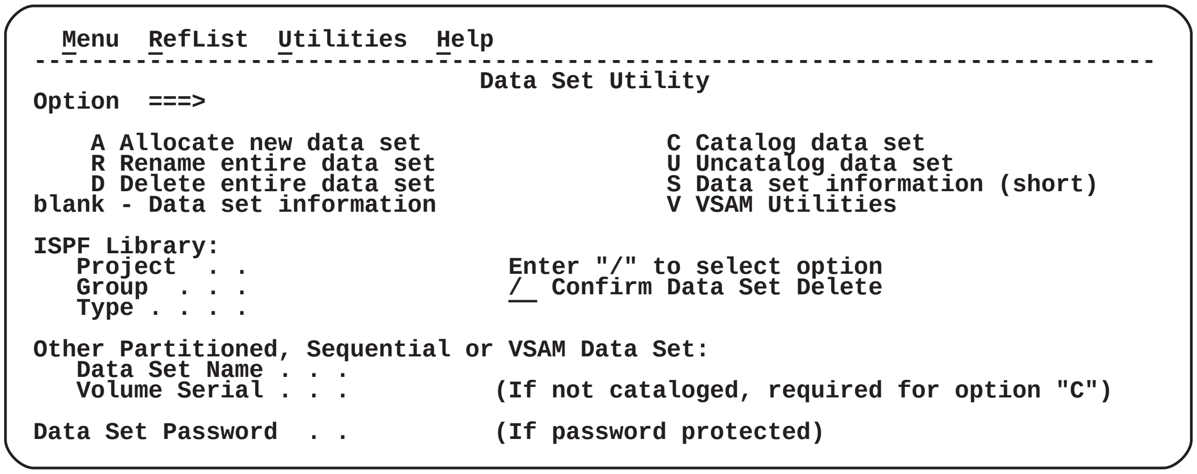
❒ **We'll be using several utilities during this class**

Finding a Data Set's Properties, 3

❒ **The best utility for finding out a data set's properties is the DATA SET utility**

- ♦ **Option 2 of Utilities (therefore, option 3.2 of ISPF)**

❒ **Selecting Option 2 shows you a screen for you to fill in the blanks regarding a) what service you want to perform and b) what data set you want to work with:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility
Option  ===>

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

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

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .          (If not cataloged, required for option "C")

Data Set Password  . .          (If password protected)
```

❒ **If you blank out the field labled "Confirm Data Set Delete", you will not get a warning prompt when you request a delete**

❒ **Typically, in the OPTION field you specify one of the choices, then fill in the name of a data set, then press Enter**

- ♦ **Since this is a common process across many ISPF functions, let's take a short digression to explore how to enter data set names in ISPF panels ...**

Entering Data Set Names in ISPF

❒ **On most panels where you are expected to enter a data set name you have two choices**

❒ **The first one looks like this:**

```
ISPF Library:
   Project  . .
   Group  . . .
   Type . . . .
   Member . . .
```

- ♦ **On some panels, the 'Member' line won't be present (for example, the Data Set utility panel)**

❒ **If the data set you're after has a three-level name, you can fill in the blanks:**

```
ISPF Library:
   Project  . . party
   Group  . . . planner
   Type . . . . list
   Member . . .
```

- ♦ **And ISPF will know you want the data set called 'PARTY.PLANNER.LIST'**
  - ✗ If the data set is a PDS and you also fill in the member field, you will access that member
  - ✗ If the data set is a PDS and you do not fill in the member field, you may get a member list to examine
  - ✗ If the data set is not a PDS, do not fill in the **Member** field

Entering Data Set Names, 2

❒ **If the data set you want does not have a name that follows ISPF naming conventions, use the second area on the panel:**

```
Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
       Name . . . _____________________________________ +
```

❒ **Fill in the entire data set name:**

```
Name . . . DEPT056.PD.OWL88.NORTY.DATA
```

♦ **ISPF will add your TSO user id to the front of this name**

✗ To avoid this, code the name in single quotes:

```
Name . . . 'APP023.LOW.TEST55.DATA2'
```

Notes

♦ **On some panels, “VSAM” will not be included in the list of possible data set types (for example, Edit cannot work with VSAM data sets unless an installation option is chosen)**

♦ **On some panels, "z/OS UNIX file" will not be included (for example, Move/Copy does not work with these; z/OS 1.9 and later)**

♦ **Generally speaking, input fields are designated with leader dots with no trailing colon (. . .)**

✗ Except for using the arrow (**===>**) on command / option lines

♦ **Display (read-only) fields typically have leader dots and a colon (. . . :)**

Entering Data Set Names, 3

❒ **If ISPF finds a name in both places, it uses the name in the lower part of the display**

```
ISPF Library:
   Project  . . party
   Group  . . . planner
   Type . . . . list
   Member . . .

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
         Name . . . _____________________________________ +
```

❒ **At any rate, after filling in this lower portion of the panel with a data set name, if the data set is a PDS, you may see a member selection list; if it is a z/OS UNIX directory, you may see a file list**

❒ **If the data set is a PDS and you know which member you want to process, you can code the member name in parentheses:**

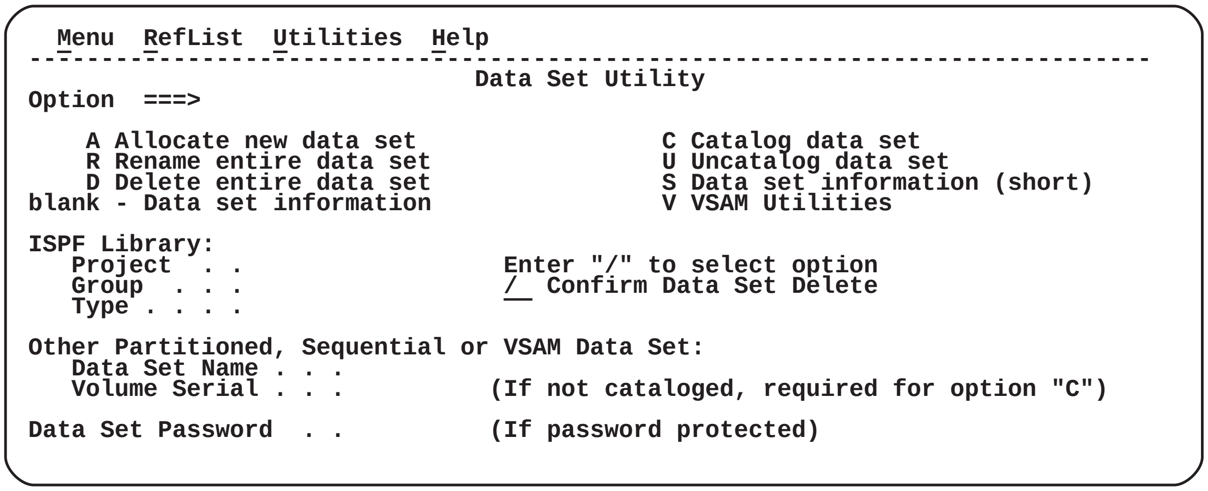
```
Name . . . LIB.COBOL(CALCWIT)
```

- ♦ **Here, ISPF will access the member 'CALCWIT' in the PDS named 'userid.LIB.COBOL'**

❒ **Data set names placed in the upper portion are remembered across ISPF sessions, while values in the lower portion are only temporary**

Finding a Data Set's Properties, continued

❐ **Recall the Data Set Utility's entry screen looks like this:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility
Option  ===>

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

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

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .          (If not cataloged, required for option "C")

Data Set Password  . .          (If password protected)
```

❐ **You can select a blank for an option (just don't enter anything there) or 'S' to get data set information**

- **Enter the data set name as discussed, then press the Enter key**

❐ **The output you get will vary depending on these properties**

- **The data set is sequential, PDS, or PDSE**
- **The data is SMS-managed or not**
- **The request is for full (blank) or short ('S') information**

❐ **In any event, ISPF will display the information you asked for**

Finding a Data Set's Properties, concluded

- Here are a couple of possible displays, depending on the type of data set and type of display requested:

Short information for non-SMS-managed PDS

```
                             Data Set Information
Command ===>

Data Set Name  . . . : SYS1.LINKLIB

General Data                          Current Allocation
 Volume serial . . . : BC3RES          Allocated blocks  . : 4,412
 Device type . . . . : 3390            Allocated extents . : 1
 Organization  . . . : PO              Maximum dir. blocks : 800
 Record format . . . : U
 Record length . . . : 0
 Block size  . . . . : 32760          Current Utilization
 1st extent blocks . : 4412            Used blocks . . . . : 2,056
 Secondary blocks  . : 0               Used extents  . . . : 1
                                       Used dir. blocks  . : 618
 Creation date . . . : 1999/03/23      Number of members . : 3,617
 Referenced date . . : 1999/07/26
 Expiration date . . : ***None***
```

Short information for SMS-managed PDS

```
                             Data Set Information
Command ===>

Data Set Name . . . . : SCOMSTO.TRAIN.SOURCE

General Data                           Current Allocation
 Management class . . : STDB            Allocated blocks  . : 61
 Storage class  . . . : STANDARD        Allocated extents . : 1
  Volume serial . . . : BCT3SX
  Device type . . . . : 3390
 Data class . . . . . :                Current Utilization
  Organization  . . . : PO              Used blocks . . . . : 5
  Record format . . . : FB              Used extents  . . . : 1
  Record length . . . : 80
  Block size  . . . . : 32720
  1st extent blocks . : 61
  Secondary blocks  . : 19
  Data set name type  : PDS

  Creation date . . . : 2002/04/20      Referenced date . . : 2002/07/26
  Expiration date . . : ***None***
```

Computer Exercise: Finding a Data Set's Properties

Get into ISPF and experiment using utilities by obtaining properties of the following data sets:

Data set name	DSORG	Volume Serial	Device type	Record length	Data class	Space allocated	Space used
________.TRAIN.LIBRARY							
________.TRAIN.ZINPUTA							
SYS1.LINKLIB							
SYS1.PROCLIB							

Get both short and long information of these files, to compare the outputs. Fill in the table above as completely as you can. Not all fields will necessarily display values.

❒ **Note: if a volume serial shows with a plus sign (*e.g.*: TSOWK1+), the data set is a multi-volume data set; pressing <Enter> will display a pop-up that lists subsequent volume serial numbers**

Section Preview

❒ **Allocating Data Sets**

- ♦ **Reserving Space**
- ♦ **Allocating a Data Set Using ISPF**
- ♦ **Allocating New Data Sets (Machine Exercise)**

Reserving Space

- **In many computer systems, to create a new file on disk you just start typing in data, or just run a program to build the file**

- **In z/OS, however, you must first reserve space for the file, a process we call allocating the file**

    - **This is because the z/OS operating system wants to control DASD space carefully, as a valuable resource**

    - **So you first describe a new file to z/OS, using ISPF or, later, JCL**

        - In terms of the properties, especially the **name**, the **device type**, and the amount of **space** you think the data set will need

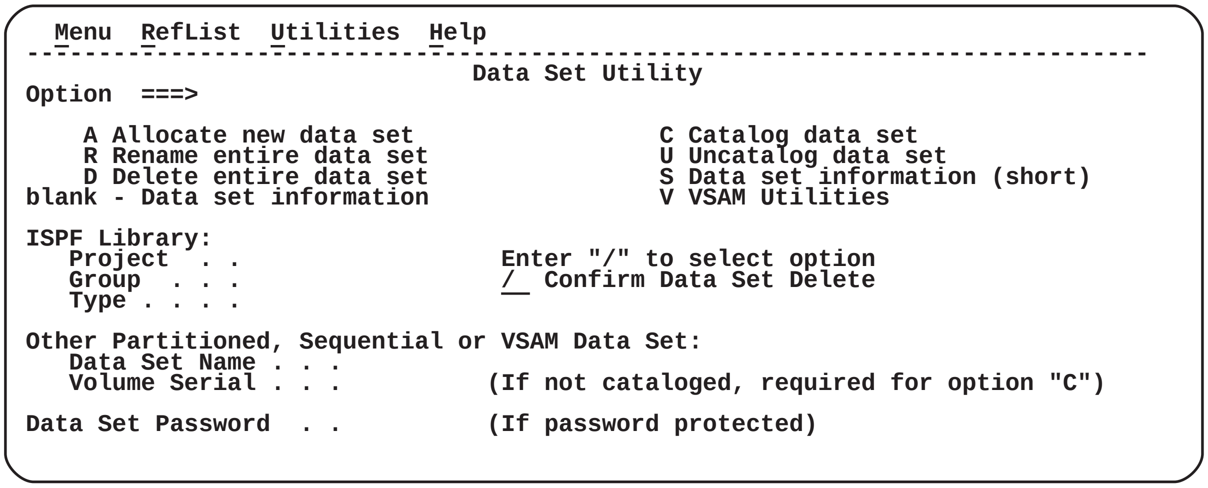
            - This initial space estimate is called a primary allocation

            - You can also request a secondary allocation: if we fill up the primary amount and need more, how much more space should we get?

Allocating a Data Set Using ISPF

❒ The same utility panel we used for examining a data set's properties can be used to allocate a new data set: ISPF option 3.2 (Utilities / Data set option), or option 2 from the Utilities pull-down

- ♦ The initial screen, recall, is:



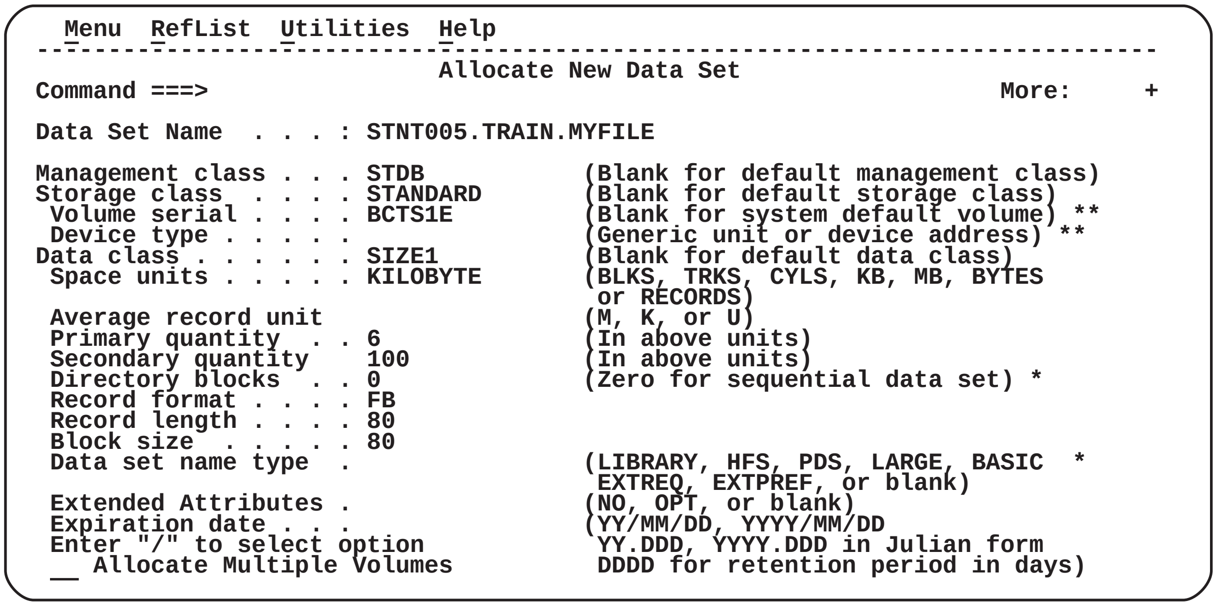
```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility
Option  ===>

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

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

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .         (If not cataloged, required for option "C")

Data Set Password  . .         (If password protected)
```

❒ Here, you select option 'A' and fill in the name of the new data set

- ♦ You will get a fill-in-the-blanks panel for specifying the properties of the new data set ...

Allocating a New Data Set

❒ **If you indicated you wanted to allocate a data set named "STNT005.TRAIN.MYFILE", by specifying option 'A', you would see a screen like this:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                            Allocate New Data Set
Command ===>                                                     More:     +

Data Set Name  . . . : STNT005.TRAIN.MYFILE

Management class . . . STDB           (Blank for default management class)
Storage class  . . . . STANDARD       (Blank for default storage class)
 Volume serial . . . . BCTS1E         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . . SIZE1          (Blank for default data class)
 Space units . . . . . KILOBYTE       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 6              (In above units)
 Secondary quantity    100            (In above units)
 Directory blocks  . . 0              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 80
 Data set name type  .                (LIBRARY, HFS, PDS, LARGE, BASIC  *
                                       EXTREQ, EXTPREF, or blank)
 Extended Attributes .                (NO, OPT, or blank)
 Expiration date . . .                (YY/MM/DD, YYYY/MM/DD
 Enter "/" to select option            YY.DDD, YYYY.DDD in Julian form
 __ Allocate Multiple Volumes          DDDD for retention period in days)
```

♦ **The actual choices will vary depending on your installation configuration options**

♦ **Typically, any values pre-filled in are just values from the last time you did an allocate or displayed some data set's properties**

♦ **You change the values as necessary to reflect the characteristics of the data set you are allocating**

♦ **By forcing Management class, Storage class, and Data class fields to be blank, you indicate you wish to allocate a non-SMS managed data set**

✗ However, installation routines may override your request, depending on local standards

Allocating a New Data Set, 2

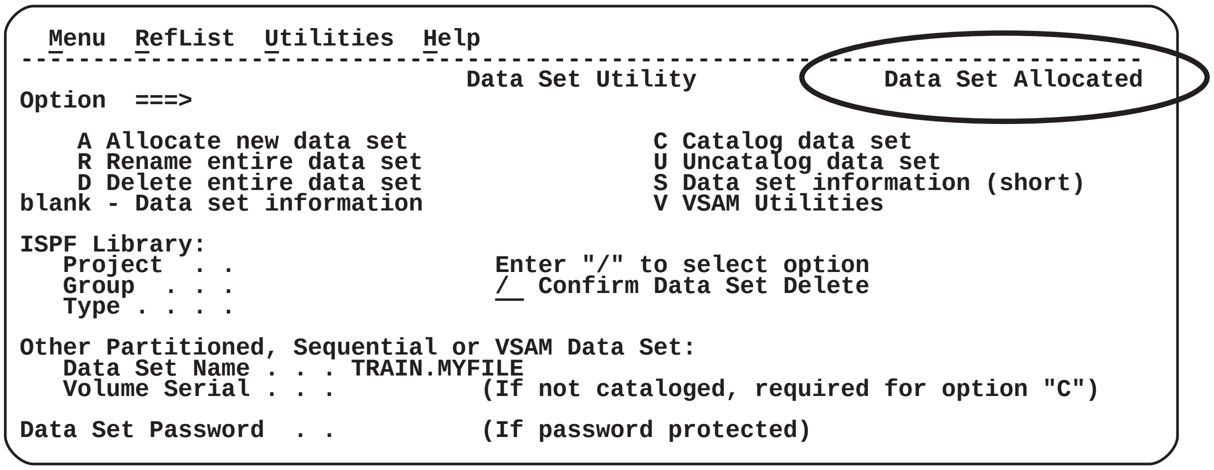
Notes

- **Make the VOLUME SERIAL blank, and specify GENERIC UNIT as an installation-dependent value, or leave blank**
- **The space units can be spelled out or have abbreviations used: specify the kinds of space to deal in**
  - ✗ Can be specified using the shortest unique abbreviation: T for tracks, C for cylinders, K for KB, M for MB, BY for bytes, R for records, and BL for blocks
- **The primary and secondary values indicate how many of these space units to allocate**
- **Specify directory blocks as non-zero for a PDS or PDSE or zero for any other kind of data set**
- **If you specify BLOCK SIZE as zero (0), the system will calculate the optimal block size for you (if you supply the record length and a record format that is not "U")**
- **The DSNTYPE field is blank or specifies one of**
  - ✗ LIBRARY - a PDSE
  - ✗ PDS - a PDS
  - ✗ HFS - space to hold Hierarchical File System files (z/OS UNIX) files
  - ✗ LARGE - large format: may have more than 65,535 tracks / volume
  - ✗ BASIC - classic format: sequential or partitioned, not LARGE, not extended format
  - ✗ EXTREQ - extended format required (VSAM only)
  - ✗ EXTPREF - extended format preferred (VSAM or sequential only)
- **The Extended Attributes field specifies if a new data set is eligible to be allocated in Extended Address Space - past cylinder 65,520; OPT means yes (z/OS 1.11)**

Allocating a New Data Set, 3

❐ **After filling in the values you want, simply press the Enter key**

- **Control will return to the previous screen with a message in the short message area indicating if the allocation was successful or not:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility             Data Set Allocated
Option  ===>

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

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

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . . TRAIN.MYFILE
   Volume Serial . . .         (If not cataloged, required for option "C")

Data Set Password  . .         (If password protected)
```

❐ **If you wanted to allocate an SMS-managed data set, you simply code the correct value(s) for** Management class**,** Storage class**, and** Data class

- **Again, installation routines may override your request**
- **Remember that the following parameter values may be implied from the Data class (so you only need to specify them on the allocate panel to override the implicit values):**
  - ✗ Space (units, primary, secondary, directory blocks)
  - ✗ Record and block information (record format, record length, block size)
  - ✗ Data set name type
  - ✗ Expiration date

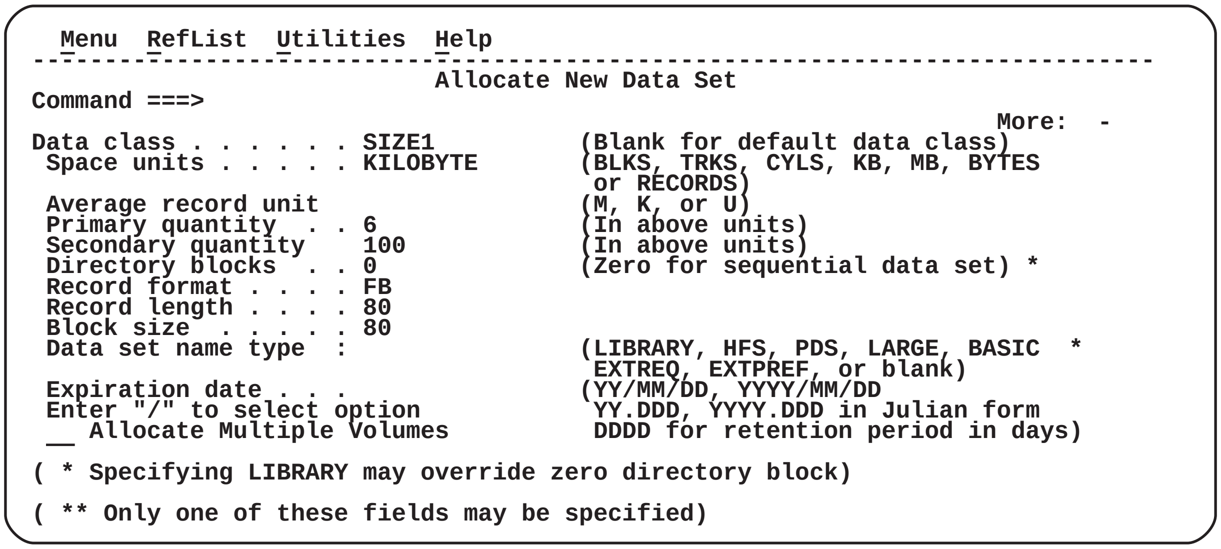
Scrollable Panels

❒ We shall look at scrolling data and lists later in the course, but notice on the previous panel, in the upper right quadrant, the string "More: +"

❒ In ISPF, when a panel cannot all fit on the screen at once, you get an indication like this to tell you there is more in the panel

- ♦ "More   +" means you can scroll the panel forward (down)
- ♦ "More   -" means you can scroll the panel backward (up)
- ♦ "More   + -" means you can scroll both ways

❒ For the panel we were just looking at, if you were to scroll down, you would see this:



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                            Allocate New Data Set
Command ===>
                                                                  More:  -
Data class . . . . . . SIZE1          (Blank for default data class)
 Space units . . . . . KILOBYTE       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 6              (In above units)
 Secondary quantity    100            (In above units)
 Directory blocks  . . 0              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 80
 Data set name type  :                (LIBRARY, HFS, PDS, LARGE, BASIC  *
                                       EXTREQ, EXTPREF, or blank)
 Expiration date . . .                (YY/MM/DD, YYYY/MM/DD
 Enter "/" to select option            YY.DDD, YYYY.DDD in Julian form
 __ Allocate Multiple Volumes          DDDD for retention period in days)

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)
```

- ♦ When the hidden information is not relevant to the discussion, we won't bother to show it (and may even omit the "More" notice)

Installation-Specific SMS-Parameter Values

❒ **Notes on <u>Management Classes</u>:**

- ♦
- ♦
- ♦

❒ **Notes on <u>Data Classes</u>:**

- ♦
- ♦
- ♦

❒ **Notes on <u>Storage Classes</u>:**

- ♦
- ♦
- ♦

❒ **Hint: look for a Primary Option Menu of ISMF or issue the ISMF command and explore from there**

Allocating a Data Set Like an Existing Data Set

- ❐ **It's often useful to create a data set just like an existing data set**
  - ♦ **Or at least, very similar**

- ❐ **This might be used to hold a copy of data for backup**
  - ♦ **Or to move data to a larger space on disk because you are running out of room**

- ❐ **The process combines steps we have already discussed:**
  - ♦ **Use Dataset Utility (ISPF option 3.2) to obtain properties of the existing data set (use an OPTION of blank)**
    - ✗ ISPF obtains and displays the properties — and remembers them
  - ♦ **Use Dataset Utility (ISPF option 3.2) to allocate the new file (use an OPTION of 'A')**
    - ✗ The ALLOCATE NEW DATA SET panel will contain the values from the above step
    - ✗ Before pressing Enter, you may now change any properties that need adjusting for your new data set (number of directory blocks, say, or primary or secondary allocation quantities)

Computer Exercise: Allocating New Data Sets

Allocate some files you will be using in later labs, with the characteristics described below. In all cases, specify the parameters we give you and set all other parameters blank (unless your instructor tells you otherwise).

1. **<userid>.TRAIN.CNTL**; a PDS for holding JCL. Specify space units of CYL, primary 1, secondary 1, 10 directory blocks; record format of FB, lrecl of 80, blocksize of 0; Data Set Name type of PDS

2. **<userid>.TRAIN.LIBRARY**; a PDS for holding source programs and small data members;

   make it like the existing data set ________.TRAIN.LIBRARY, except only request space as 10MB primary and 2MB secondary, and only allow 5 directory blocks

3. **<userid>.TRAIN.LOAD**; a PDS for holding executable programs;

   Specify space units of CYL, primary 1, secondary 1, 5 directory blocks, record format of U, record length of 0, block size of 32760, and a Data Set Name Type of PDS

4. **<userid>.TRAIN.ZINPUTX**; a sequential data set; specify a space units of TRK, primary 1, secondary 1; record length of 100, record format of FB, block size of 0, and directory blocks 0

-- more --

Computer Exercise: Allocating New Data Sets, p. 2.

5. **<userid>.TRAIN.INPUTA**; a sequential data set; specify a record length of 100, block size of 0, record format of FB, and directory blocks 0; ask for space as 2 tracks primary, no secondary.

In all cases, replace "<userid>" by your TSO id.

Next, examine the properties of these data sets and see if any of them have had SMS properties assigned to them; if so, note them down here for discussion after the lab.

Dataset	Dataclass	Management class	Storage class
CNTL	________	____________	____________
LIBRARY	________	____________	____________
LOAD	________	____________	____________
ZINPUTX	________	____________	____________
INPUTA	________	____________	____________

❐ **Note: if you place a slash ("/") next to the Allocate Multiple Volumes option near the bottom of the Allocate panel
(see page 101), you will see a pop-up for you to allocate up to 20 volumes**

- ♦ **DO NOT DO THAT HERE: this is just for your information**

Section Preview

❐ **Looking at Data — Edit, View, and Browse**

- ♦ **Edit, View, and Browse**
- ♦ **Notes on View and Edit Entry Panels**
- ♦ **Member Selection Lists**
- ♦ **More Notes on View and Edit Entry Panels**
- ♦ **Some View / Edit Commands**
- ♦ **Introduction to 'View' (Machine Exercise)**

Edit, View, and Browse

❐ **ISPF has several options for viewing, updating, and changing data**

Edit - allows users to

- ♦ **Add to existing data**
- ♦ **Key in data where there is none**
- ♦ **Change / manipulate data, using line commands and primary commands**
- ♦ **Save the revised version for later work**
- ♦ **Requires exclusive use of data being edited**

View - allows users to

- ♦ **Add to existing data**
- ♦ **Key in data where there is none**
- ♦ **Change / manipulate data, using line commands and primary commands**
- ♦ **Needs only shared access to data being edited**

❐ **View cannot save changes except by copying changed data to another file**

- ♦ **View has an option called Browse, used in earlier versions of ISPF, that has a much more restricted set of capabilities**
- ♦ **View replaces Browse; it's like Edit without update ability**
  - ✗ You can request Browse mode on entry to View

Edit, View, and Browse, continued

- ❒ **In this course, we will demonstrate how to get into both View and Edit**
  - ♦ **Including how to invoke Browse mode**

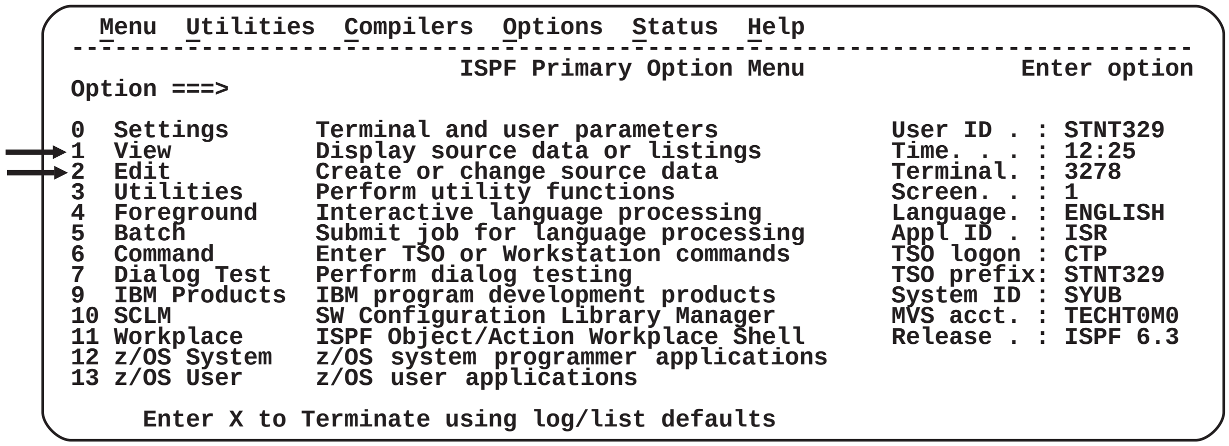
- ❒ **Then we will focus on Edit**
  - ♦ **Pointing out where View behaves differently**

- ❒ **You get to View from Option 1 of the Primary Option Menu**
  - ♦ **Or from Option 2 of the Menu Action bar choice**

- ❒ **You get to Edit from Option 2 of the Primary Option Menu**
  - ♦ **Or from Option 3 of the Menu Action bar choice**

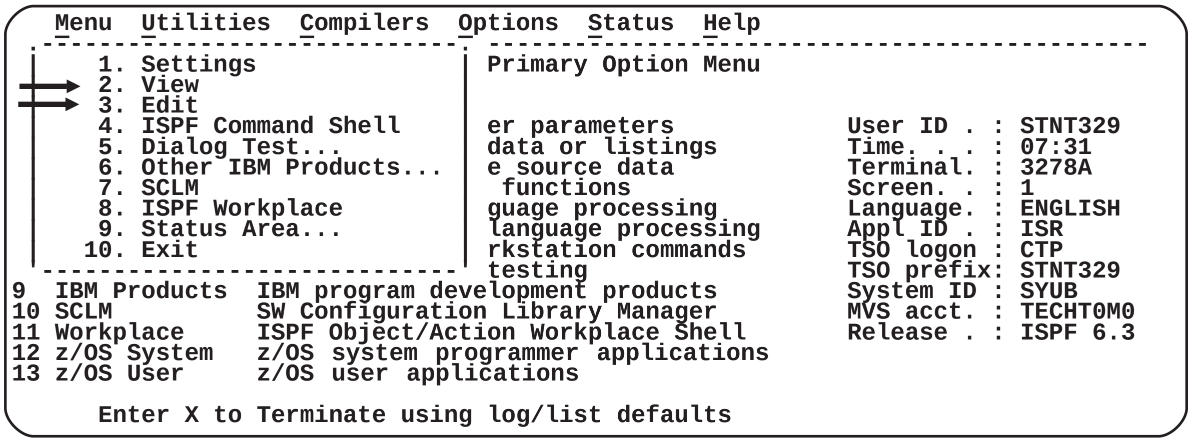
Edit, View, and Browse, continued

❐ **From the Primary Option Menu ...**



```
  Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                           ISPF Primary Option Menu              Enter option
 Option ===>

 0  Settings      Terminal and user parameters          User ID . : STNT329
 1  View          Display source data or listings       Time. . . : 12:25
 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 : CTP
 7  Dialog Test   Perform dialog testing                TSO prefix: STNT329
 9  IBM Products  IBM program development products      System ID : SYUB
 10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
 11 Workplace     ISPF Object/Action Workplace Shell     Release . : ISPF 6.3
 12 z/OS System   z/OS system programmer applications
 13 z/OS User     z/OS user applications

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

❐ **From the Menu Action bar choice:**

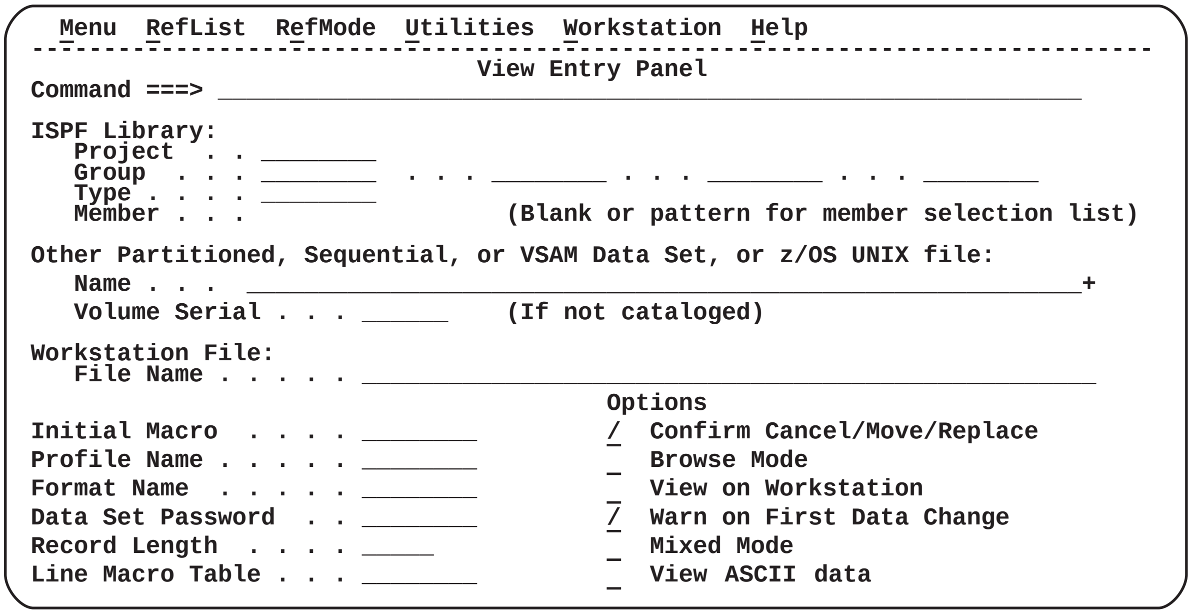


```
   Menu  Utilities  Compilers  Options  Status  Help
  .------------------------------. ---------------------------------------------
  |     1. Settings              | Primary Option Menu
  |     2. View                  |
  |     3. Edit                  |
  |     4. ISPF Command Shell     | er parameters          User ID . : STNT329
  |     5. Dialog Test...        | data or listings       Time. . . : 07:31
  |     6. Other IBM Products... | e source data          Terminal. : 3278A
  |     7. SCLM                  |  functions             Screen. . : 1
  |     8. ISPF Workplace        | guage processing       Language. : ENGLISH
  |     9. Status Area...        | language processing    Appl ID . : ISR
  |    10. Exit                  | rkstation commands     TSO logon : CTP
  '------------------------------' testing                TSO prefix: STNT329
 9  IBM Products  IBM program development products      System ID : SYUB
 10 SCLM          SW Configuration Library Manager      MVS acct. : TECHT0M0
 11 Workplace     ISPF Object/Action Workplace Shell     Release . : ISPF 6.3
 12 z/OS System   z/OS system programmer applications
 13 z/OS User     z/OS user applications

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

❐ **Remember, if you use the Action bar, you nest functions; if you use options from the menu, you are using the hierarchy**

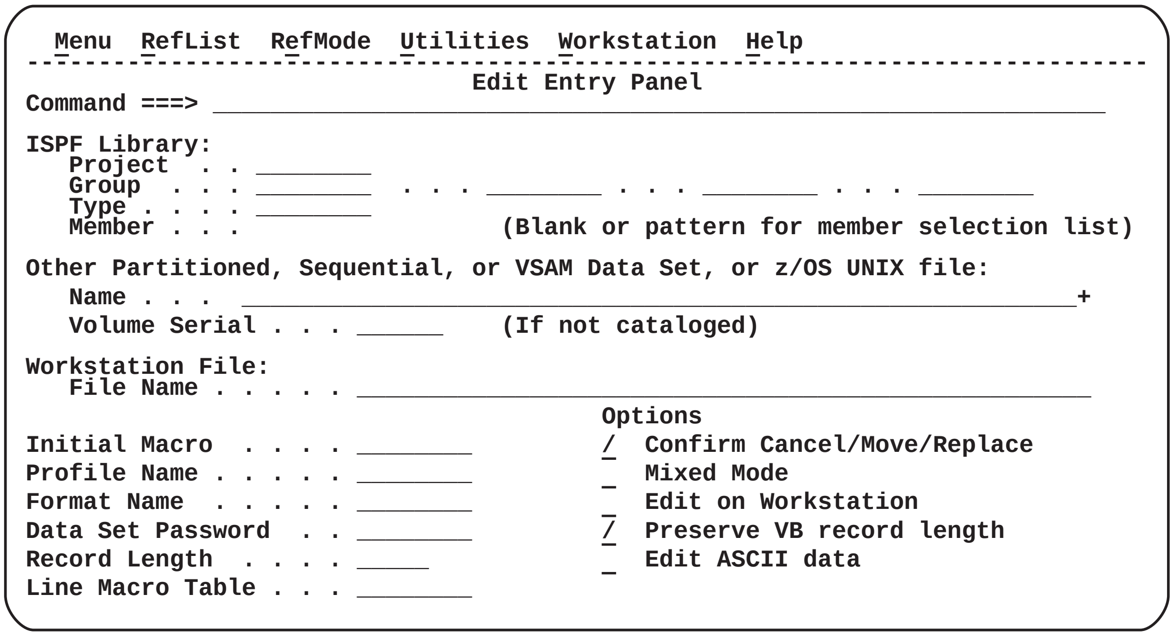
Edit, View, and Browse, continued

❒ **The View Entry panel looks like this:**



```
  Menu  RefList  RefMode  Utilities  Workstation  Help
-------------------------------------------------------------------------------
                            View Entry Panel
Command ===> ____________________________________________________________

ISPF Library:
   Project  . . ________
   Group  . . . ________  . . . ________ . . . ________ . . . ________
   Type . . . . ________
   Member . . .                  (Blank or pattern for member selection list)

Other Partitioned, Sequential, or VSAM Data Set, or z/OS UNIX file:
   Name . . .  __________________________________________________________+
   Volume Serial . . . ______    (If not cataloged)

Workstation File:
   File Name . . . . . ___________________________________________________
                                      Options
Initial Macro  . . . . ________        /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________        _  Browse Mode
Format Name  . . . . . ________        _  View on Workstation
Data Set Password  . . ________        /  Warn on First Data Change
Record Length  . . . . _____           _  Mixed Mode
Line Macro Table . . . ________        _  View ASCII data
```

❒ **The Edit Entry panel looks like this:**



```
  Menu  RefList  RefMode  Utilities  Workstation  Help
-------------------------------------------------------------------------------
                            Edit Entry Panel
Command ===> ______________________________________________________________

ISPF Library:
   Project  . . ________
   Group  . . . ________  . . . ________ . . . ________ . . . ________
   Type . . . . ________
   Member . . .                  (Blank or pattern for member selection list)

Other Partitioned, Sequential, or VSAM Data Set, or z/OS UNIX file:
   Name . . .  __________________________________________________________+
   Volume Serial . . . ______    (If not cataloged)

Workstation File:
   File Name . . . . . ___________________________________________________
                                      Options
Initial Macro  . . . . ________        /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________        _  Mixed Mode
Format Name  . . . . . ________        _  Edit on Workstation
Data Set Password  . . ________        /  Preserve VB record length
Record Length  . . . . _____           _  Edit ASCII data
Line Macro Table . . . ________
```

❒ **There are very few differences**

- ♦ **We examine the options on the following pages**

Notes on View and Edit Entry Panels

❒ **To View or Edit, from the entry panel ...**

- ♦ **Specify the name of the data to be Viewed or Edited:**

  - ✗ Enter a <u>sequential data set</u> name:

    - ➢ Placed into full screen view or edit of sequential data set

  - ✗ Or, enter a <u>PDS or PDSE data set</u> name <u>and member name</u>:

    - ➢ Placed into full screen view or edit of the member

  - ✗ Or, enter a <u>PDS or PDSE data set</u> name with <u>no member name</u>, or a <u>pattern for a member name</u>:

    - ➢ Shown member list panel
    - ➢ Move cursor next to desired member name, type in 'S' and press <Enter> key:
    - ➢ Placed into full screen view or edit of selected member

  - ✗ Or enter the name of an HFS path (begin with a slash (/) or a tilde(~)) [z/OS 1.9 or later]

    - ➢ If this is a file name, placed into edit or view of the file
    - ➢ If this a directory name, a directory selection list is displayed
    - ➢ HFS files are only discussed peripherally in this course

- ♦ **You may specify the name of a file on your workstation**

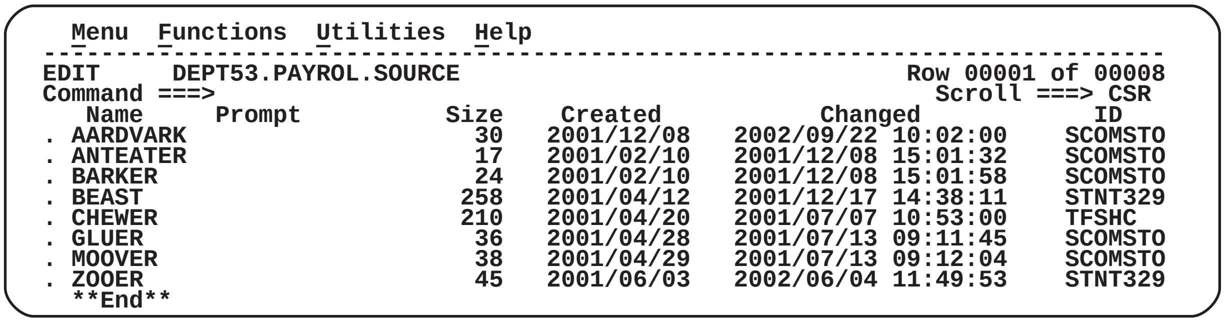
    - ➢ Not discussed here

- ♦ **You may view or edit an ASCII file (ISPF converts to and from)**

Member Selection Lists

❒ For many functions, you may need to see the list of members in a PDS

- ♦ If so, ISPF displays a 'member selection list' panel
- ♦ Although the format varies a little depending on the function, there is always a Command field followed by the member name field
- ♦ Member lists are always sorted by name initially
- ♦ For example:



```
  Menu  Functions  Utilities  Help
------------------------------------------------------------------------------
EDIT     DEPT53.PAYROL.SOURCE                              Row 00001 of 00008
Command ===>                                                 Scroll ===> CSR
   Name     Prompt         Size    Created          Changed          ID
. AARDVARK                   30   2001/12/08   2002/09/22 10:02:00    SCOMSTO
. ANTEATER                   17   2001/02/10   2001/12/08 15:01:32    SCOMSTO
. BARKER                     24   2001/02/10   2001/12/08 15:01:58    SCOMSTO
. BEAST                     258   2001/04/12   2001/12/17 14:38:11    STNT329
. CHEWER                    210   2001/04/20   2001/07/07 10:53:00    TFSHC
. GLUER                      36   2001/04/28   2001/07/13 09:11:45    SCOMSTO
. MOOVER                     38   2001/04/29   2001/07/13 09:12:04    SCOMSTO
. ZOOER                      45   2001/06/03   2002/06/04 11:49:53    STNT329
  **End**
```

- ♦ Note also, in the upper right corner (in the short message area) is information regarding how many members there are and which member name is at the top of the visible list
- ♦ The format of this list varies a little from release to release, but the general information stays the same

Patterns In Member Lists

- **On some panels, not coding a member name produces a list of all members; on other panels a member name may be specified as '*'**
  - **Which means 'ALL MEMBERS'**

- **A member name may also sometimes be specified as a pattern**
  - **A partial name that includes one or more pattern characters:**

'*'	**—**	**Any number of characters with any value**
'%'	**—**	**A single character with any value**

Examples

Member . . . *mod* **might result in the list**

CMODA
CMODB
CMODC
C2MODA
C2MODB
C2MODC
MODULEE
REMODEL

Member . . . a*d%2 **might find**

AAD02
ABCD12

Member Selection List Panels

❒ **If member list longer than screen depth ...**

- ♦ **You may issue the scroll commands <u>UP</u> and <u>DOWN</u> to scroll the list:**

```
            Command ===> up
   Or       Command ===> down
```

- ♦ **You may issue the <u>LOCATE</u> command:**

Command ===> locate *member-name*

✗ **The list is positioned so the named member is at the top**

➢ If the named member does not exist, you are positioned to the member which would immediately precede the named member

✗ **You can use this feature to position yourself in a long list:**

```
Command ===> locate xr
```

✗ **This scrolls the list so the first member name beginning with 'XR' is near the top of the screen**

✗ **'LOCATE' may be abbreviated 'LOC' or just 'L'**

Member Selection List Panels, 2

❒ **To select a member from an Edit or View member selection list**

- ♦ **Move the cursor next to the desired member name (use the 'NEW LINE' key)**
- ♦ **Type in the letter 'S', and press <Enter>**
  - ✗ Typing in a slash ('/') works the same as 'S'
  - ✗ You are placed into Edit or View of the member

Or

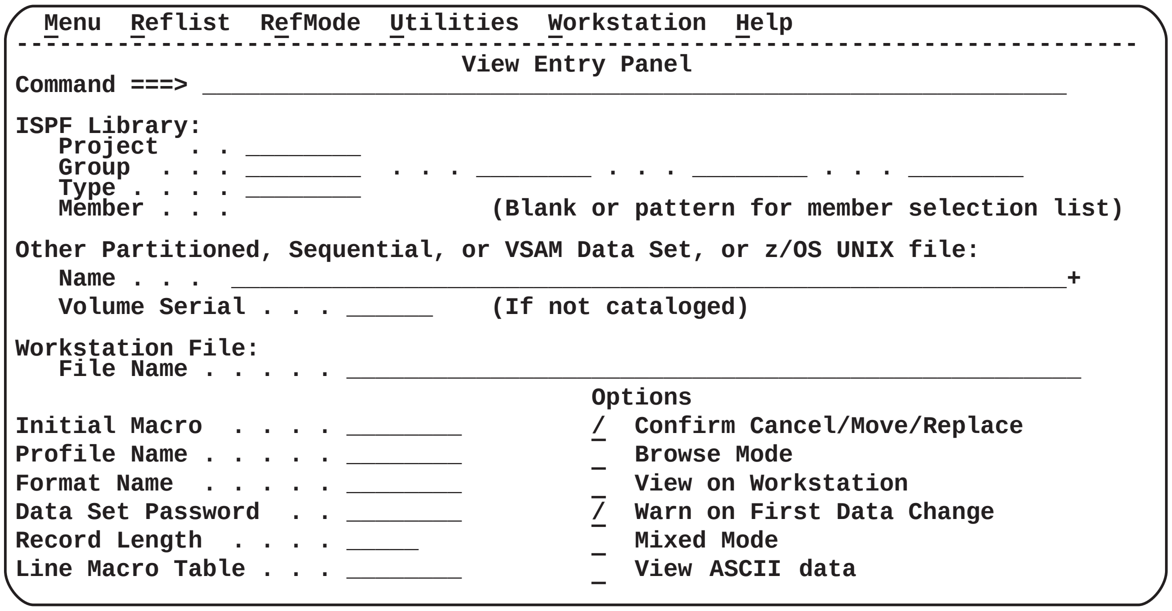
- ♦ **Issue the SELECT command from the command / option line:**

```
Command ===> select member-name
```

- ♦ **'SELECT' may be abbreviated 'SEL' or 'S'**

More Notes on View and Edit Entry Panels

❒ Using the View Entry panel as a model ...



```
  Menu  Reflist  RefMode  Utilities  Workstation  Help
------------------------------------------------------------------------------
                               View Entry Panel
Command ===> ____________________________________________________________

ISPF Library:
   Project  . . ________
   Group  . . . ________  . . . ________ . . . ________ . . . ________
   Type . . . . ________
   Member . . .                  (Blank or pattern for member selection list)

Other Partitioned, Sequential, or VSAM Data Set, or z/OS UNIX file:
   Name . . .  __________________________________________________________+
   Volume Serial . . . ______    (If not cataloged)

Workstation File:
   File Name . . . . . ___________________________________________________
                                        Options
Initial Macro  . . . . ________         /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________         _  Browse Mode
Format Name  . . . . . ________         _  View on Workstation
Data Set Password  . . ________         /  Warn on First Data Change
Record Length  . . . . _____            _  Mixed Mode
Line Macro Table . . . ________         _  View ASCII data
```

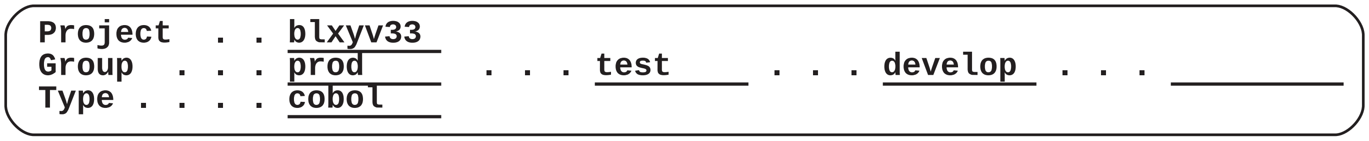
- ♦ **We've already discussed placing the name of the object you want to view or edit or browse**

- ♦ **But look at the "Group" line:**

```
Group  . . . ________  . . . ________ . . . ________ . . . ________
```

✗ You specify multiple "Group" values if concatenating a list of data sets with a similar name structure

➢ For example ...



```
Project  . . blxyv33
Group  . . . prod     . . . test     . . . develop  . . . ________
Type . . . . cobol
```

➢ Indicates you want to work with members in three libraries:

BLXYV33.PROD.COBOL
BLXYV33.TEST.COBOL
BLXYV33.DEVELOP.COBOL

Even More Notes on View and Edit Entry Panels

- **The name field in the middle of the panel is a scrollable field (HFS file names can be up to 1024 characters long)**
    - **If the cursor is in the field, scroll left and right keys scroll the contents of the field, not the panel**
- **“Volume Serial” and “Data Set Password” are rarely used (not at all in this class)**
- **“Initial Macro” is used to identify a small program (edit macro) you wish to run against the data before you even see it**
    - **Rarely used (not at all in this class)**
- **“Profile Name” identifies a named collection of attributes (for example: if data should be forced to upper case or not; should sequence numbers be saved with the data) you want to work under during this view / edit session**
    - **Discussed in detail later**
- **“Format Name” is used to support working with Double Byte Character Set (DBCS) data (such as Japanese kanji characters on special terminals)**
    - **Not discussed in this class**
- **"Record Length" is used when viewing or editing z/OS UNIX files: format the data so it looks like fixed length records of this length**
    - **When editing, on exit the file is converted to a fixed length file of this record length (the value you specify must be at least as large as the largest record actually in the file or you get an error)**
- **"Line Macro Table" specifies where to find installation-defined macros that can be used as line commands - not discussed further (z/OS 1.13 and later)**

Still More Notes on View and Edit Entry Panels

- ❒ **On the right hand / center portion of the entry panel are options you can select**

  - ♦ **Place a slash ("/") next to an option to select it**

    - ✗ Selecting “Browse Mode” invokes the old Browse panel

    - ✗ Selecting “Confirm Cancel/Move/Replace” ensures you will get a warning message to prevent inadvertent deletion of data

    - ✗ Selecting “Mixed Mode”, like the Format Name parameter, is for support of DBCS data

    - ✗ Select "Warn on First Data Change" (View only) to get a message if you actually change the data

    - ✗ Selecting "Preserve VB record length" (Edit only) ensures trailing blanks don't get deleted when you save a VB file

      - ➢ On the command line, you may also enter "**PRESERVE ON**" or "**PRESERVE OFF**"

- ❒ **On the Edit entry panel you also have the choice of specifying the name of a workstation file; basically, you can**

  - ♦ **Edit workstation files on the workstation (using the workstation editor of your choice)**

  - ♦ **Edit mainframe files on the workstation (using the workstation editor of your choice; the file is downloaded to start and uploaded on end)**

  - ♦ **Edit workstation files on the mainframe (using the ISPF editor; the file is uploaded to start and downloaded on end)**

  - ♦ **Edit mainframe files on the mainframe (using the ISPF editor)**

Action Bar Choices on View and Edit Entry Panels

❒ **Both these panels have these choices:**

- **Menu - the same choices as from the Menu choice on the Primary Option Menu (see p. 112)**

- **RefList and RefMode - used to implement referral lists**

  - ✗ Storing lists of data sets, for quick reference in view, edit, and utilities functions

  - ✗ Discussed later

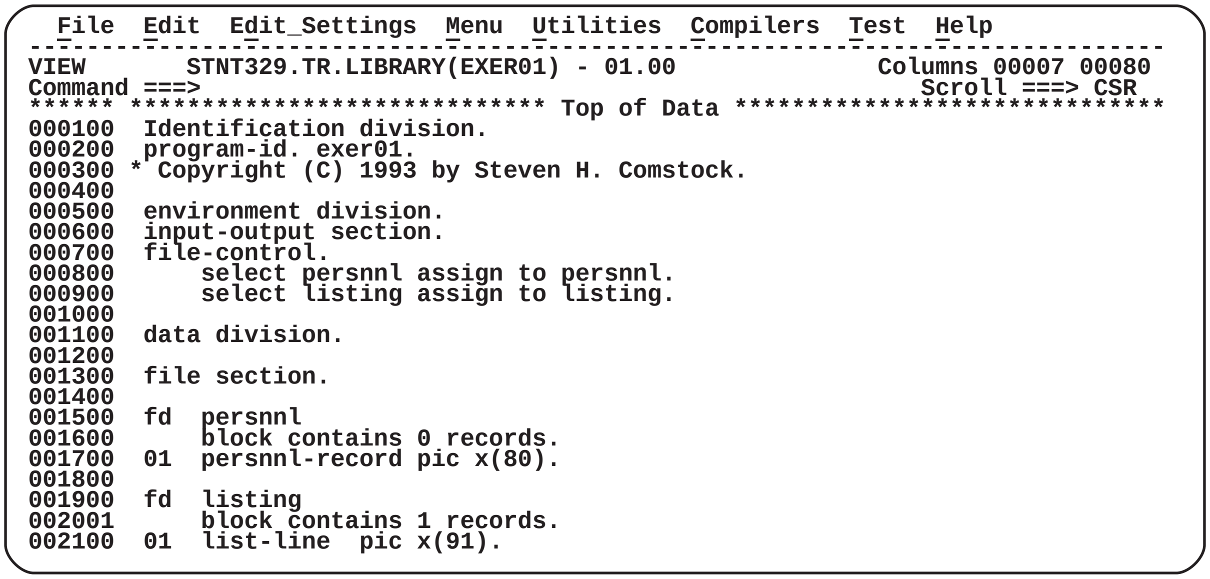
- **Utilities - the same choices as from the Utilities choice on the Primary Option Menu (see p. 90)**

- **Help - various topics that may be of interest as you work**

- **Workstation - a choice for supplying workstation related information**

View / Edit Panels

❒ **When you are in View of a data set or member, the panel looks like this:**



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
VIEW       STNT329.TR.LIBRARY(EXER01) - 01.00              Columns 00007 00080
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100  Identification division.
000200  program-id. exer01.
000300 * Copyright (C) 1993 by Steven H. Comstock.
000400
000500  environment division.
000600  input-output section.
000700  file-control.
000800      select persnnl assign to persnnl.
000900      select listing assign to listing.
001000
001100  data division.
001200
001300  file section.
001400
001500  fd  persnnl
001600      block contains 0 records.
001700  01  persnnl-record pic x(80).
001800
001900  fd  listing
002001      block contains 1 records.
002100  01  list-line  pic x(91).
```

❒ **Regarding the action bar choices**

- ♦ **We'll discuss File, Edit, and Edit__Settings later**
- ♦ **Menu, Utilities, and Help are the same choices we've seen before**
- ♦ **Compilers and Test are not discussed in this course**

❒ **The Edit panel is the same except the third line says EDIT instead of VIEW**

- ♦ **If you enter in Browse mode, the title says BROWSE and there are no sequence numbers**

Some View / Edit Commands

❒ When you are in view or edit, you may issue several commands at the command line level, including:

<u>LOCATE line-number</u>

- ♦ Position data at this line number at the top of the screen ('line-number' must be an integer)
- ♦ Note that “LOCATE 0” positions you at the top, and “LOCATE 999999” positions you at the bottom
- ♦ LOCATE may be abbreviated LOC or just L, and upper and lower case are equivalent (you may key “locate”, “loc”, or “l”)

<u>COLS [ON | OFF]</u>

- ♦ Display column numbers at the top of the display area; omitting ON and OFF toggles this setting

<u>RESET</u>

- ♦ Remove columns line from the display, in Browse mode

<u>HEX [ON | OFF]</u>

- ♦ Display data in hexadecimal or character format

❒ Remember that 'END' leaves the data being displayed and returns to the previous screen, while 'RETURN' goes back to the Primary Option Menu

Computer Exercise: Introduction to 'View'

Get into ISPF/PDF and perform the following tasks:

Select the View option and ...

1. View the PDS named ________.TRAIN.LIBRARY
   - + How many members does it have? ______

2. In this library, select the member called INPEDTA by placing an 's' next to the member name in the member list

3. Locate line number 227 and copy down the contents of this line below:

____

4. End your View of INPEDTA, returning to the member list

5. Use the SELECT command to View member INPUT1. Can you make any sense of this data?

6. Return to the View entry screen

7. View the sequential data set INPUTX
   (full name: ________.TRAIN.INPUTX)

   This file contains non-character data, such as binary numbers, so parts of the display will look a little strange.

   How many records (lines) are there in this file? ______

Exit ISPF/PDF and logoff.

Section Preview

❐ **More on Edit and View, and Help**

- ♦ **More Edit / View commands**
- ♦ **String types**
- ♦ **Help**
- ♦ **More Edit and View, and Help (Machine Exercise)**

More Edit / View Commands

❒ **In addition to the commands already discussed:**

LOCATE line-number

COLS

RESET

HEX

END

RETURN

❒ **Edit and View support additional commands, including**

FIND string

RFIND

UP

DOWN

LEFT

RIGHT

FIND - Edit / View Primary Command

```
FIND  string    [NEXT]    [CHARS]    [col-1]    [col-2]
                [ALL]     [PREFIX]
                [FIRST]   [SUFFIX]
                [LAST]    [WORD]
                [PREV]
```

string	—	**String to search for**
NEXT	—	**Begin at cursor location and scan forward (down)**
ALL	—	**Find all occurrences of the string in the file**
FIRST	—	**Go to the top and search down to find the first**
LAST	—	**Go to the bottom and search up to find the last**
PREV	—	**Begin at cursor location and scan backward (up)**
CHARS	—	**Locate these characters as standalone or embedded in words**
PREFIX	—	**Only look for these characters at the start of a word**
SUFFIX	—	**Only look for these characters at the end of a word**
WORD	—	**Only look for these characters as a standalone word**
col-1	—	**Left column boundary to restrict search**
col-2	—	**Right column boundary to restrict search**

Strings

- ❒ The FIND, CHANGE, and EXCLUDE Edit and View commands all reference 'strings'

- ❒ You may use any one of five kinds of strings:

Simple string

- ♦ A string not starting or ending with an apostrophe or quotation mark, and without embedded blanks, commas, or asterisks

Delimited string

- ♦ Any string bounded by apostrophes but not containing any embedded apostrophes, or bounded by quotes but not containing any embedded quotes

Character string

- ♦ A delimited string preceded or followed by a 'C'

Hexadecimal string

- ♦ Any delimited string of valid hex digits preceded or followed by a character 'X'

Picture string

- ♦ Any delimited string of picture characters preceded or followed by the letter 'P'

Strings - Examples

Simple strings

MASTER-FILE

Any

- **Searches using simple strings are not case sensitive:**

FIND Any

- **will result in a match with each of these values in the data:**

ANY	ANy	AnY	Any
aNY	aNy	anY	any

Delimited strings

'HERE ARE EMBEDDED BLANKS, COMMAS, AND ** ASTERISKS'

"HERE WE HAVE EMBEDDED 'APOSTROPHES'"

'HERE WE HAVE EMBEDDED "QUOTES"'

- **Searches using delimited strings are not case sensitive**

- **You must use a delimited string if the string both contains and uses commands or keywords (*e.g.*: find all first)**

Strings - Examples, 2

Character strings

C'This is the most common form of character strings'

'BUT THIS WORKS ALSO'C

- ♦ **Searches using character strings ARE case sensitive:**

C'Any'

- ♦ **will only match with 'Any'**

Hexadecimal strings

- ♦ **The valid characters in a hex string are A - F, a - f, 0 - 9**

X'0000457C'

'FF00001C994040'X

Strings - Examples, 3

Picture strings

- **Identify types of characters in positions**

Picture character	Meaning
=	any character
¬	any non-blank character
.	any non-displayable character
#	any numeric character, 0 - 9
-	any non-numeric character
@	any alphabetic character, any case
<	any lowercase alphabetic character, a - z
>	any uppercase alphabetic character, A - Z
$	any special character (non-alphanumeric)

- **Picture strings can include alphanumeric characters, which represent themselves [in a non-case-sensitive way]**

❒ **Describe the kind of data being searched for with these picture strings**

```
P'###'

P'¬   ¬'

P'@@  #'

P'PART###@@'
```

FIND - Examples

```
FIND     PENGUINS

FIND     ALL          PENGUINS

FIND     PENGUINS        ALL        30    71

FIND     'Peace of mind'  FIRST

FIND     'trouble'C       NEXT

FIND     X'402021'        PREV

FIND     'END'            SUFFIX     20    60

FIND     "LINCOLN'S DOCTOR'S DOG"   ALL

FIND     P'M<'        ALL
```

❒ **Note that you can use an unquoted asterisk as the find string, and the last string used will be used again; for example:**

```
find    penguins    first
f   *   last
```

RFIND: Repeat Find

Syntax

RFIND

Function

Repeat last FIND command

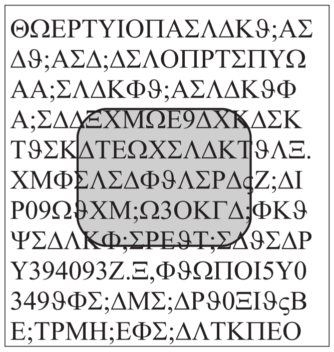
- **Direction of the search is implied by the previous FIND command**
  - **If FIND specified NEXT, ALL, FIRST, direction is forward (downward) in the file**
  - **If FIND specified LAST or PREV, direction is backward (upward) in the file**

- **RFIND is usually assigned to F5 / F17 when you are in Edit or View**

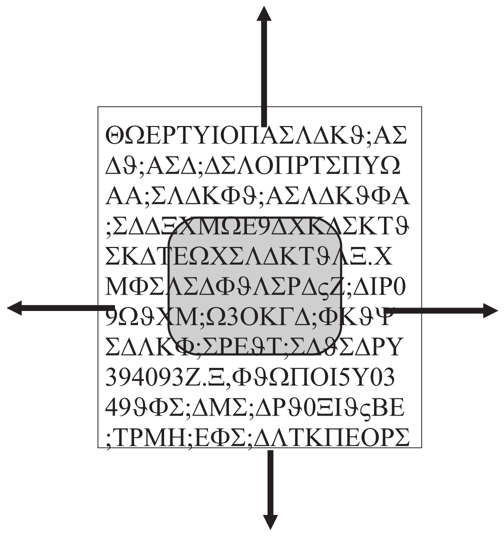
Scrolling

When data to be displayed or edited is larger than the screen size, ISPF supports a scrolling function

- ♦ **Think of the data as a large sheet of paper, and the display as a window looking out over a portion of the data**

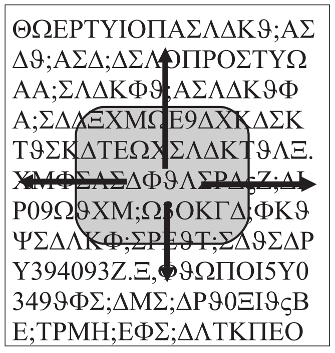


- ♦ **You can think of scrolling either as moving the paper while the window stays put:**



- ♦ **Or moving the window while the data stays put:**

  ISPF uses this approach



Scrolling Commands

❒ **Scrolling is implemented by four commands**

UP	**-**	**Scroll towards top of data**
DOWN	**-**	**Scroll towards bottom of data**
LEFT	**-**	**Scroll towards left edge of data**
RIGHT	**-**	**Scroll towards right edge of data**

❒ **ISPF also uses FORWARD as a synonym of DOWN, and BACKWARD as a synonym of UP**

❒ **In z/OS 1.5, the concept of a <u>scrollable field</u> was introduced**

- ♦ **A text field that does not display all of the contents of a related data item**

❒ **When the cursor is on a scrollable field and a LEFT or RIGHT scroll command is issued, the scroll is within the field only**

❒ **Then, for all situations, the only question is, "How far should we scroll?" ...**

Scroll Amount

❒ **The amount of data to scroll may be any of these:**

PAGE	-	A screen's worth
CURSOR (CSR)	-	Position to the cursor (if the cursor is on the command line, this works the same as 'PAGE')
DATA	-	A screen's worth less one line (so you have one line 'overlap')
HALF	-	Half a screen's worth
MAX	-	All the way UP / DOWN / LEFT / RIGHT
nnnn	-	The number of lines or columns to scroll

♦ **The amount may be coded after the command,**
e.g.: **UP MAX**

♦ **But if no amount is coded, the value is taken from the scroll amount field in the upper right hand corner of the screen**

✗ **You may change this amount by tabbing over and typing the scroll amount yourself; this value will stay there until you change it again**

♦ **Note that ISPF provides TOP as a synonym for "UP MAX" and BOTTOM as a synonym for "DOWN MAX"**

✗ MAX may be abbreviated M (or m)

Some Function Key Settings

❒ **ISPF provides you with some standard function key assignments**

✗ Initially, we care about these:

Function keys	**Assigned value**
1 / 13	HELP
3 / 15	END / EXIT
4 / 16	RETURN
5 / 17	RFIND
7 / 19	UP / BACKWARD
8 / 20	DOWN / FORWARD
10 / 22	LEFT
11 / 23	RIGHT
12 / 24	CANCEL

♦ **If you enter a value in the command line and then press a function key, the system takes the function key string followed by the command line string**

♦ **Thus typing in 'm' on the command line and pressing F8, you are in essence issuing the command 'DOWN m', that is, go to the bottom of the file**

More on Function Keys

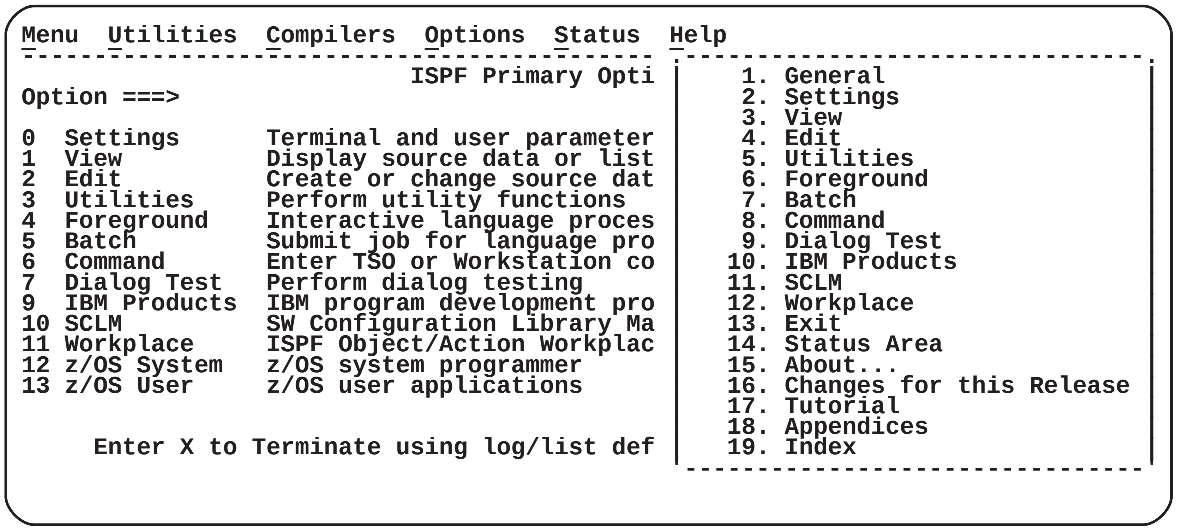
- ❐ **Key assignments change as you move about ISPF from function to function**
  - ♦ **For example, if you are in an environment where scrolling isn't appropriate or necessary (at the Primary Option Menu, for example), the scrolling keys are not assigned**
- ❐ **Whenever possible, however, ISPF has tried to assign keys consistently across environments, so some key assignments are always the same ...**
  - ♦ ***e.g.:* F1 is always Help**
- ❐ **... or at least equivalent function**
  - ♦ ***e.g.:* F3: End and Exit are both 'leaving' kinds of commands**

Help

❒ **ISPF has an extensive facility of embedded Help**

- ♦ **Whenever you question a message, request Help (F1)**
- ♦ **Every Action Bar has a Help choice**

❒ **From the Primary Option Menu, the Help pull-down menu looks like this:**



```
Menu  Utilities  Compilers  Options  Status  Help
----------------------------------------------- .--------------------------------.
                           ISPF Primary Opti |     1. General                 |
Option ===>                                  |     2. Settings                |
                                             |     3. View                    |
0  Settings      Terminal and user parameter |     4. Edit                    |
1  View          Display source data or list |     5. Utilities               |
2  Edit          Create or change source dat |     6. Foreground              |
3  Utilities     Perform utility functions   |     7. Batch                   |
4  Foreground    Interactive language proces |     8. Command                 |
5  Batch         Submit job for language pro |     9. Dialog Test             |
6  Command       Enter TSO or Workstation co |    10. IBM Products            |
7  Dialog Test   Perform dialog testing      |    11. SCLM                    |
9  IBM Products  IBM program development pro |    12. Workplace               |
10 SCLM          SW Configuration Library Ma |    13. Exit                    |
11 Workplace     ISPF Object/Action Workplac |    14. Status Area             |
12 z/OS System   z/OS system programmer      |    15. About...                |
13 z/OS User     z/OS user applications      |    16. Changes for this Release|
                                             |    17. Tutorial                |
                                             |    18. Appendices              |
     Enter X to Terminate using log/list def |    19. Index                   |
                                             '--------------------------------'
```

❒ **As you can see, there is lots of information available, including a Tutorial for general learning on your own ...**

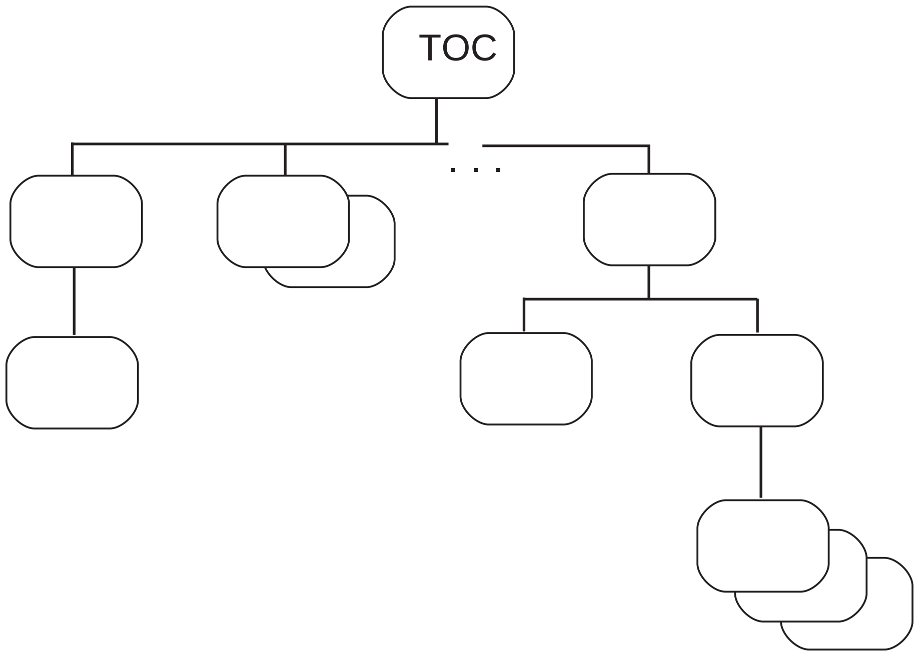
Tutorial

❐ **Explanation of terms**

❐ **Guidance in coding commands and parameters**

❐ **Always has a Table of Contents (TOC) at the head of the chain**

❐ **May have an index**



❐ **Pressing <Enter> steps you through the tutorial panels sequentially, from top to bottom, left to right**

Tutorial Commands

BACK	-	Back up one screen in the tutorial
SKIP	-	Skip this topic
UP	-	Display higher level of topical list
TOC	-	Go to the Table of Contents
INDEX	-	Go to the alphabetical index of topics available

<u>Tutorial Function key settings</u>

1 / 13	-	HELP
3 / 15	-	END
7 / 19	-	UP (display higher level of topical list)
8 / 20	-	SKIP to next topic
10 / 22	-	Backup to previous page
11 / 23	-	Move forward to next page

✗ The <Enter> key is also interpreted as "next page"

❒ **Use END or RETURN to exit the tutorial**

Computer Exercise - More on Edit and View, and Help

Get into ISPF/PDF and perform the following tasks:

Select the View option and ...

1. Work with the library named ________.TRAIN.LIBRARY

2. In this library, select the member called INPEDTA

   a. locate line number 170; copy down the contents of this line in the space below:

____

   b. How many occurrences are there of the string "LOVE"?

      no. of occurrences: ___; line number of first:____

      line numbers of last 3 occurrences: ___ ___ ___

   c. How many occurrences are there of the apostrophe (" ' ") character? ____

3. View the sequential data set called ________.TRAIN.ZINPUTA

   a. How many records does this data set have? ____

   b. How many occurrences are there of the string "all"? ____

4. Get out of view.

Select the Tutorial option from the Help pull-down menu

1. Find the Index and use it to look up a topic
2. Pick a topic that interests you and follow it for a few minutes
3. Use the index to find out about the panelid command, then leave the Tutorial and issue the panelid command.

Exit ISPF/PDF and logoff TSO.

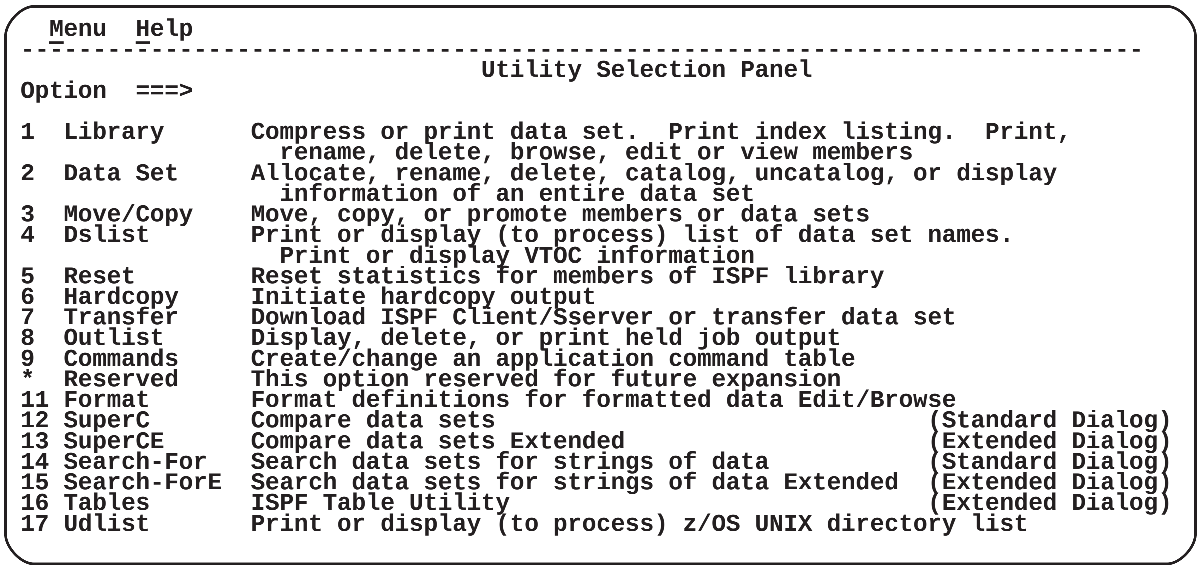
Section Preview

❐ **More Utility Functions**

- ♦ **Move / Copy**
- ♦ **Deleting a Data Set**
- ♦ **Renaming a Data Set**
- ♦ **The Library Utility**
- ♦ **Sorting Member Lists**
- ♦ **Utility Functions (Machine Exercise)**

ISPF Utilities

❒ **Recall that ISPF option 3 provides you with a list of utility functions, like this (or you can get there from the Action bar):**



```
   Menu  Help
 ------------------------------------------------------------------------------
                                Utility Selection Panel
 Option  ===>

 1  Library      Compress or print data set.  Print index listing.  Print,
                   rename, delete, browse, edit or view members
 2  Data Set     Allocate, rename, delete, catalog, uncatalog, or display
                   information of an entire data set
 3  Move/Copy    Move, copy, or promote members or data sets
 4  Dslist       Print or display (to process) list of data set names.
                   Print or display VTOC information
 5  Reset        Reset statistics for members of ISPF library
 6  Hardcopy     Initiate hardcopy output
 7  Transfer     Download ISPF Client/Sserver or transfer data set
 8  Outlist      Display, delete, or print held job output
 9  Commands     Create/change an application command table
 *  Reserved     This option reserved for future expansion
 11 Format       Format definitions for formatted data Edit/Browse
 12 SuperC       Compare data sets                                 (Standard Dialog)
 13 SuperCE      Compare data sets Extended                        (Extended Dialog)
 14 Search-For   Search data sets for strings of data              (Standard Dialog)
 15 Search-ForE  Search data sets for strings of data Extended     (Extended Dialog)
 16 Tables       ISPF Table Utility                                (Extended Dialog)
 17 Udlist       Print or display (to process) z/OS UNIX directory list
```

❒ **Up to now, we have only used some of the facilities of the DATA SET utility**

- ♦ **Display data set information**
- ♦ **Allocate data set (for standard and SMS-managed data sets)**

❒ **In this section, we explore**

- ♦ **The move / copy utility (option 3 above)**
- ♦ **More facets of the data set utility (option 2)**
- ♦ **The library utility (option 1)**

MOVE / COPY Utility (Option 3.3)

❒ **Move or copy a sequential file to a new file**

- ♦ **Space for the new file either has been previously allocated**

  - ✗ Or, if the data set does not exist, you are prompted how you want to create it

❒ **Move or copy all or some members of a library to another library**

  - ✗ Including prompting to create it if target library does not exist

❒ **Move or copy one member of a library to a sequential data set**

  - ✗ Including prompting to create it if target data set does not exist

❒ **Optionally, may request a printout at the same time a copy, move, lock, or promote is done**

<u>Notes</u>

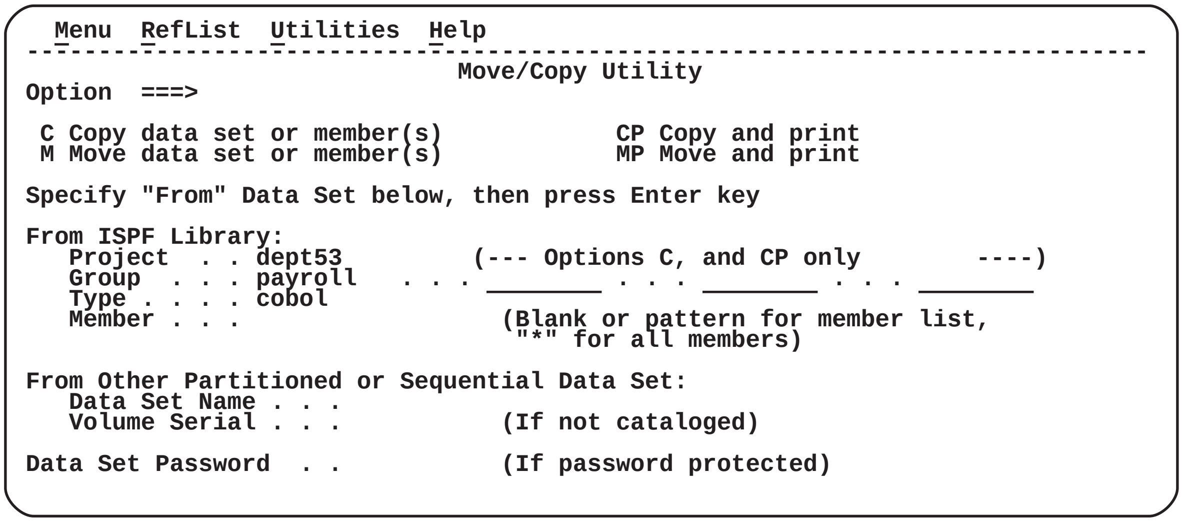
- ♦ **MOVE and COPY both copy data from the source to the target**

- ♦ **But MOVE then deletes the source (the original), while COPY leaves the original data intact**

- ♦ **The option.sub_option notation (*e.g.*: "3.3") only applies when you are using the menu hierarchy**

MOVE / COPY, Specify Source Data Set

- ❒ **On the panel you request the copy, move, lock or promote service, you also identify the FROM (source) data set**

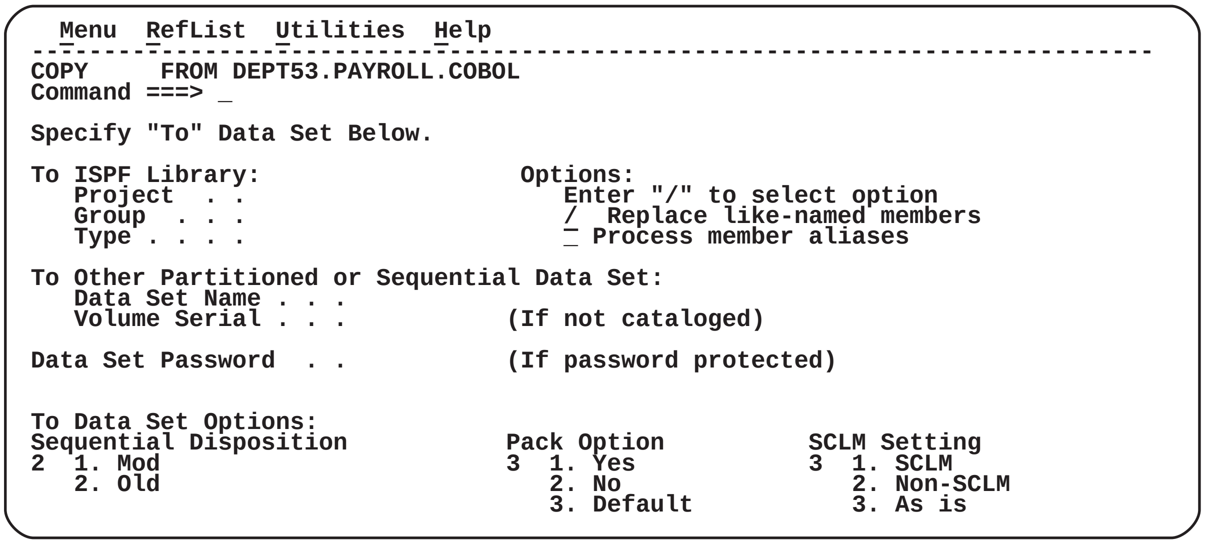


```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                              Move/Copy Utility
Option  ===>

 C Copy data set or member(s)            CP Copy and print
 M Move data set or member(s)            MP Move and print

Specify "From" Data Set below, then press Enter key

From ISPF Library:
   Project  . . dept53         (--- Options C, and CP only        ----)
   Group  . . . payroll   . . . ________ . . . ________ . . . ________
   Type . . . . cobol
   Member . . .                  (Blank or pattern for member list,
                                  "*" for all members)

From Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Data Set Password  . .           (If password protected)
```

- ❒ **First enter one of the four choices in the Option field (C, CP, M, or MP)**

- ❒ **Then fill in the From data set in one of the two areas and when you press <Enter>, you see a panel to identify the TO (target) data set, as shown on the next page**

MOVE / COPY, Specify Target Data Set



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
COPY     FROM DEPT53.PAYROLL.COBOL
Command ===> _

Specify "To" Data Set Below.

To ISPF Library:                  Options:
   Project  . .                      Enter "/" to select option
   Group  . . .                      /  Replace like-named members
   Type . . . .                      _ Process member aliases

To Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Data Set Password  . .           (If password protected)


To Data Set Options:
Sequential Disposition           Pack Option          SCLM Setting
2  1. Mod                        3  1. Yes            3  1. SCLM
   2. Old                           2. No                2. Non-SCLM
                                    3. Default           3. As is
```

Notes

- **The panel title identifies the function and the source**
- **You then specify the target and any options ...**
  - ✗ For the Replace like-named PDS members option, key a "/" if it is OK to replace members already in the target data set that have the same name as members coming from the source data set; Process aliases: beyond scope of course
  - ✗ For the Sequential Disposition option, Old says to overlay any existing data in the target sequential data set, while Mod says to add source data after any existing data in the target
  - ✗ The Pack option compresses data on DASD; normally choose No (or, Default: pack target data if source data is packed)
- **If your source and target data sets are sequential, when you press <Enter> the copy or move is done**
- **If your source and target data sets are PDSs, then you need to consider what members to move or copy ...**

MOVE / COPY, PDSs

❐ **If your FROM data set is a PDS, on the initial panel you may specify a member name**

- ♦ **To just move or copy a single member**

❐ **Or you may specify an asterisk (*)**

- ♦ **To move or copy all members**

❐ **Or you may specify a pattern**

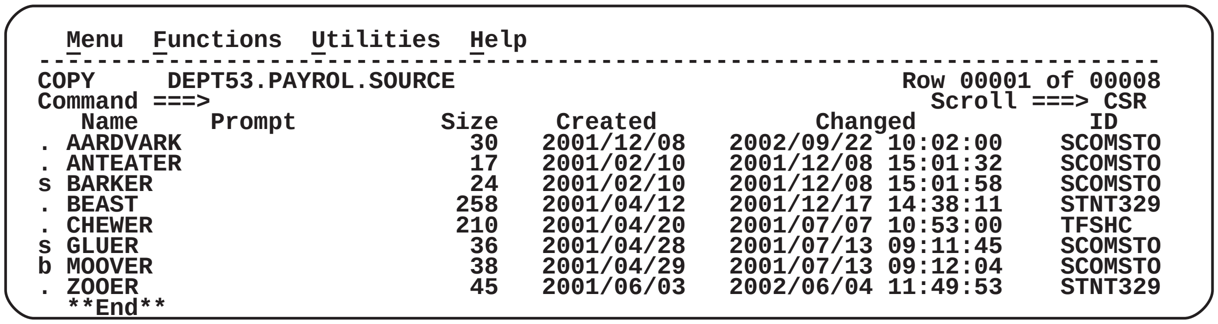
- ♦ **To get a member list of member names that satisfy the pattern**

❐ **Or you may specify a blank**

- ♦ **To get a complete member list**

❐ **If you get a member list, place an S next to the member names you want to copy or move**

- ♦ **Place a B to browse a member, so you can decide if you want to copy or move it, or not**



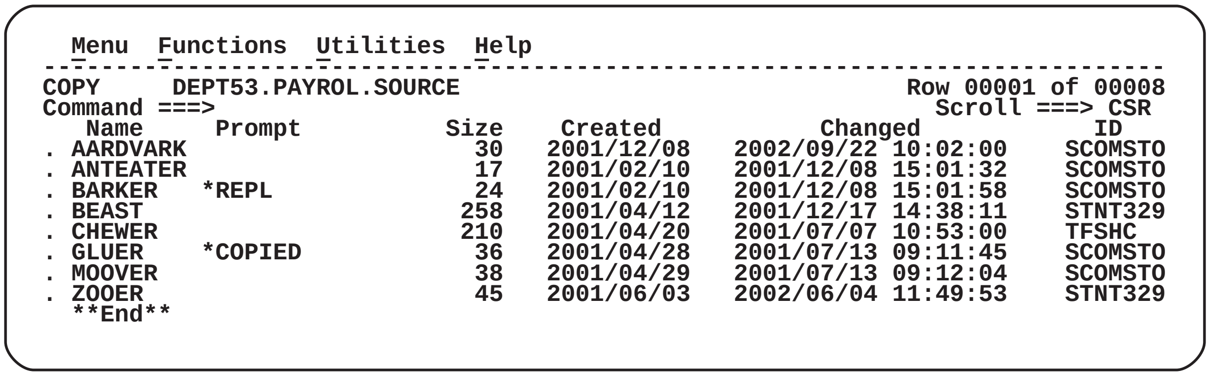
```
  Menu  Functions  Utilities  Help
------------------------------------------------------------------------------
COPY      DEPT53.PAYROL.SOURCE                               Row 00001 of 00008
Command ===>                                                   Scroll ===> CSR
   Name      Prompt          Size    Created         Changed             ID
. AARDVARK                     30   2001/12/08   2002/09/22 10:02:00    SCOMSTO
. ANTEATER                     17   2001/02/10   2001/12/08 15:01:32    SCOMSTO
s BARKER                       24   2001/02/10   2001/12/08 15:01:58    SCOMSTO
. BEAST                       258   2001/04/12   2001/12/17 14:38:11    STNT329
. CHEWER                      210   2001/04/20   2001/07/07 10:53:00    TFSHC
s GLUER                        36   2001/04/28   2001/07/13 09:11:45    SCOMSTO
b MOOVER                       38   2001/04/29   2001/07/13 09:12:04    SCOMSTO
. ZOOER                        45   2001/06/03   2002/06/04 11:49:53    STNT329
  **End**
```

- ♦ **Notice that you can rename a member as you move or copy it!**
- ♦ **Anyway, press <Enter> and the move or copy is done ...**

MOVE / COPY, PDSs, continued

❒ **After the move or copy, the PROMPT column is used to tell you what members were moved / copied**

- ♦ **If an existing member was replaced, that information is displayed, also**


```
  Menu  Functions  Utilities  Help
------------------------------------------------------------------------------
COPY     DEPT53.PAYROL.SOURCE                                Row 00001 of 00008
Command ===>                                                   Scroll ===> CSR
   Name      Prompt          Size    Created            Changed           ID
. AARDVARK                     30   2001/12/08   2002/09/22 10:02:00    SCOMSTO
. ANTEATER                     17   2001/02/10   2001/12/08 15:01:32    SCOMSTO
. BARKER    *REPL              24   2001/02/10   2001/12/08 15:01:58    SCOMSTO
. BEAST                       258   2001/04/12   2001/12/17 14:38:11    STNT329
. CHEWER                      210   2001/04/20   2001/07/07 10:53:00    TFSHC
. GLUER     *COPIED            36   2001/04/28   2001/07/13 09:11:45    SCOMSTO
. MOOVER                       38   2001/04/29   2001/07/13 09:12:04    SCOMSTO
. ZOOER                        45   2001/06/03   2002/06/04 11:49:53    STNT329
  **End**
```

Making a Backup Copy of a Data Set

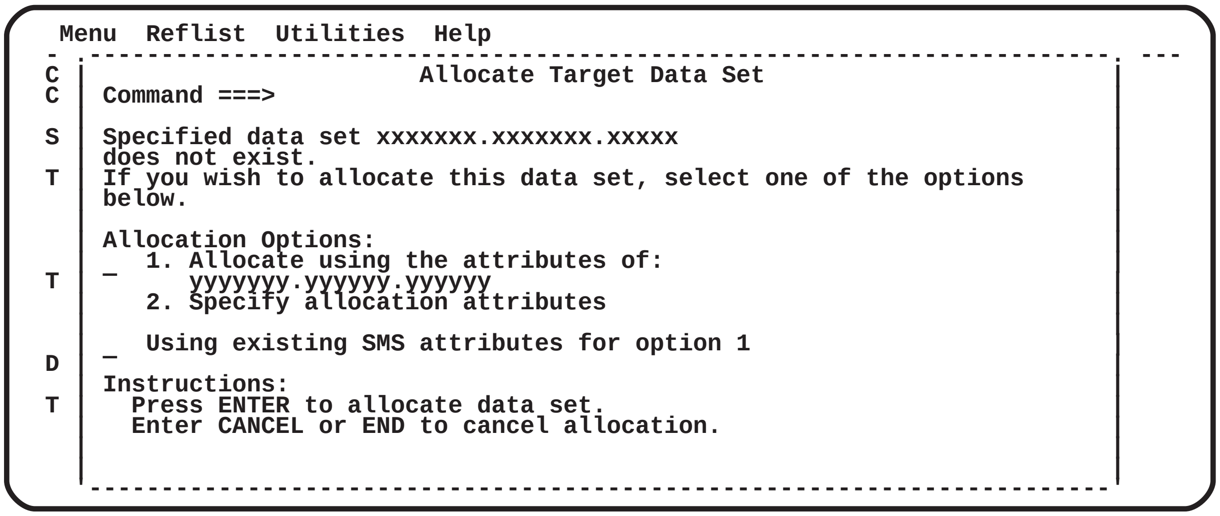
❒ Earlier, on page 106, we discussed how to create a data set just like an existing data set

- ♦ But all that did was to allocate space

❒ MOVE / COPY can complete the process of making a backup copy by using ISPF 3.3, Copy option, to actually copy the data from the original to the backup data set

- ♦ Or, of course, selecting Move/Copy from the Utilities pull-down menu

❒ Alternatively, you can take advantage of the fact that the move / copy operation itself will prompt you to allocate a file if one doesn't already exist and do it all in one step:



```
 Menu  Reflist  Utilities  Help
- .-----------------------------------------------------------------------. ---
C |                       Allocate Target Data Set                        |
C | Command ===>                                                          |
  |                                                                       |
S | Specified data set xxxxxxx.xxxxxxx.xxxxx                              |
  | does not exist.                                                       |
T | If you wish to allocate this data set, select one of the options      |
  | below.                                                                |
  |                                                                       |
  | Allocation Options:                                                   |
  | _  1. Allocate using the attributes of:                               |
T |       yyyyyyy.yyyyyy.yyyyyy                                           |
  |    2. Specify allocation attributes                                   |
  |                                                                       |
  |    Using existing SMS attributes for option 1                         |
D | _                                                                     |
  | Instructions:                                                         |
T |   Press ENTER to allocate data set.                                   |
  |   Enter CANCEL or END to cancel allocation.                           |
  |                                                                       |
  |                                                                       |
  '-----------------------------------------------------------------------'
```

- ♦ If you select option 2, you are shown the ISPF 3.2 allocate screen

Deleting a Data Set

❒ **Use the dataset utility, utility option 2 (ISPF option 3.2, then), to delete an existing data set**

- ♦ **Recall the Data Set Utility panel looks like this:**

```
  Menu  RefList  Utilities  Help
 ------------------------------------------------------------------------------
                              Data Set Utility
Option  ===>

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

ISPF Library:
   Project  . .                 Enter "/" to select option
   Group  . . .                 /_ Confirm Data Set Delete
   Type . . . .

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .         (If not cataloged, required for option "C")

Data Set Password  . .         (If password protected)
```

❒ **To delete a data set, choose option D and enter the name of the data set you want to delete**

- ♦ **You will get a confirmation panel: do you really want to do this?**

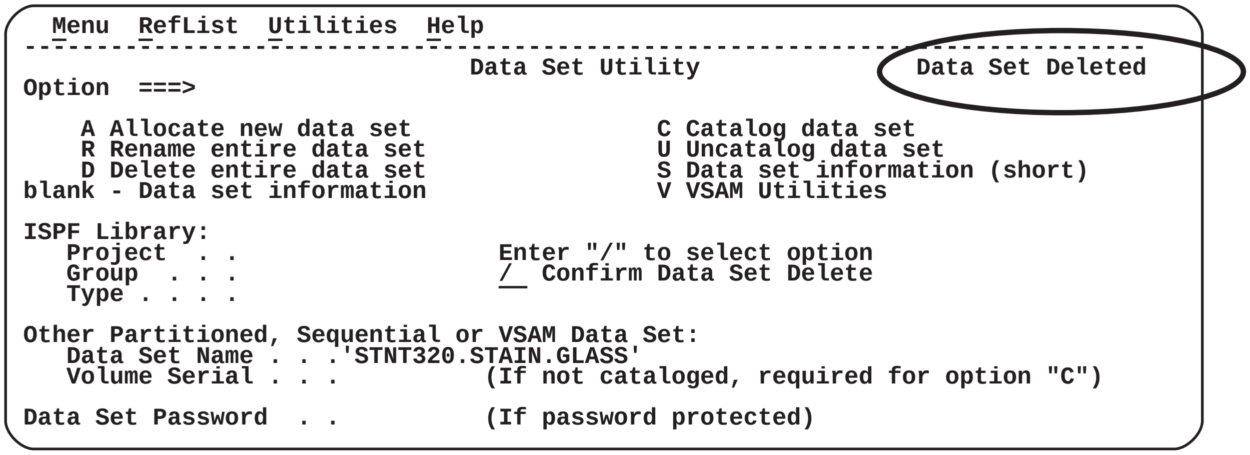
  - ✗ Note that in z/OS, if you delete a data set, it is gone; you cannot recover unless you have made a backup copy

- ♦ **The Delete confirmation panel looks like this:**

```
.------------------------  Confirm Delete  -------------------------.
|                                                                    |
| Command  ===> _                                                    |
|                                                                    |
| Data Set Name. : STNT320.STAIN.GLASS                               |
| Volume . . . . : GOOFY                                             |
| Creation Date. : 2002/05/05                                        |
|                                                                    |
| Enter "/" to select option                                         |
| _  Set data set delete confirmation off                            |
|                                                                    |
| Instructions:                                                      |
|                                                                    |
|  Press ENTER key to confirm the delete request.                    |
|  (The data set will be deleted and uncataloged.)                   |
|                                                                    |
|  Press CANCEL or EXIT to cancel the delete request.                |
|                                                                    |
|                                                                    |
|                                                                    |
|                                                                    |
'--------------------------------------------------------------------'
```

Deleting a Data Set, continued

❒ **If you press <Enter>, the data set is deleted and you are returned to the previous panel with a confirmation message in the short message area:**

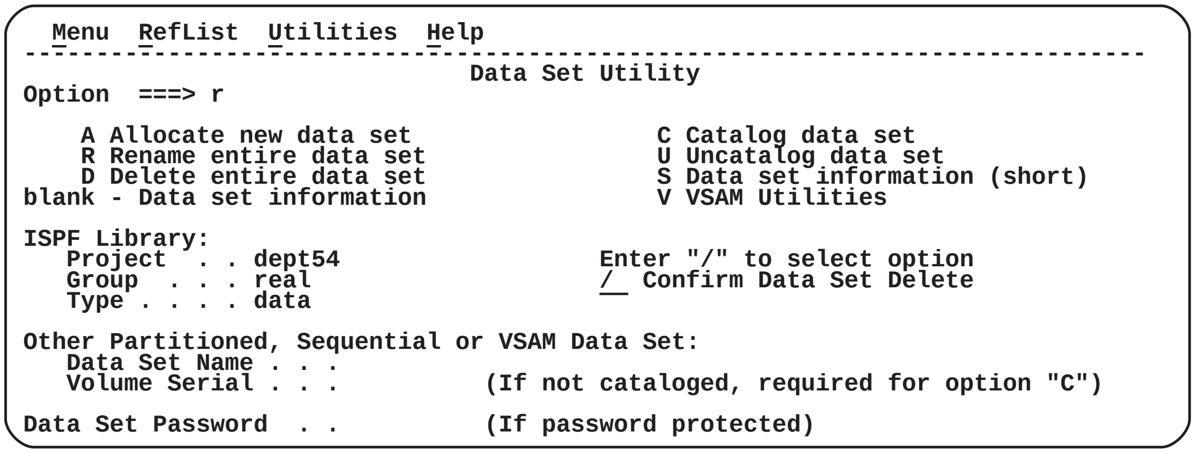


```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility               Data Set Deleted
Option  ===>

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

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

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .'STNT320.STAIN.GLASS'
   Volume Serial . . .          (If not cataloged, required for option "C")

Data Set Password  . .          (If password protected)
```

Renaming a Data Set

❒ **Use the data set utility, utility option 2 (ISPF option 3.2, then), to rename an existing data set**

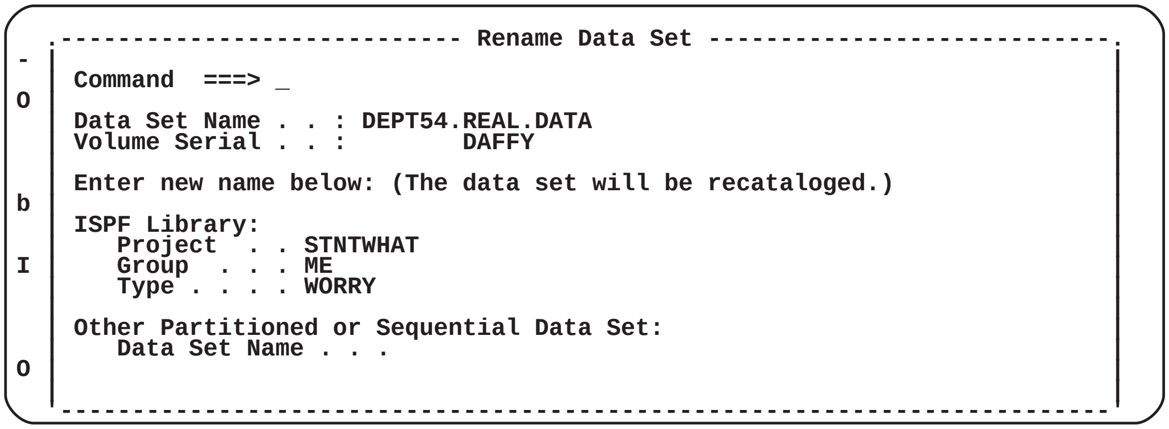
- **From the Data Set Utility panel, select option R and enter the current name of the data set you want to rename:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Data Set Utility
Option  ===> r

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

ISPF Library:
   Project  . . dept54                   Enter "/" to select option
   Group  . . . real                     /  Confirm Data Set Delete
   Type . . . . data

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .          (If not cataloged, required for option "C")

Data Set Password  . .          (If password protected)
```

- **You will get a rename panel: what do you want the new name to be?**

- **The rename panel looks like this:**



```
   .--------------------------- Rename Data Set ----------------------------.
-  |                                                                        |
   | Command  ===> _                                                        |
O  |                                                                        |
   | Data Set Name . . : DEPT54.REAL.DATA                                   |
   | Volume Serial . . :        DAFFY                                       |
   |                                                                        |
   | Enter new name below: (The data set will be recataloged.)              |
b  |                                                                        |
   | ISPF Library:                                                          |
   |    Project  . . STNTWHAT                                               |
I  |    Group  . . . ME                                                     |
   |    Type . . . . WORRY                                                  |
   |                                                                        |
   | Other Partitioned or Sequential Data Set:                              |
   |    Data Set Name . . .                                                 |
O  |                                                                        |
   |                                                                        |
   '------------------------------------------------------------------------'
```

- **Key in the new name, press <Enter>, and you have it**

Renaming Caution

- **Security packages often allow you to designate specific volumes as eligible to only hold data sets with certain naming patterns**

  - **For example, perhaps volumes LIB001 through LIB005 are only allowed to hold data sets that begin with "DEPT53", "DEPT55", and "ACCNTNG"**

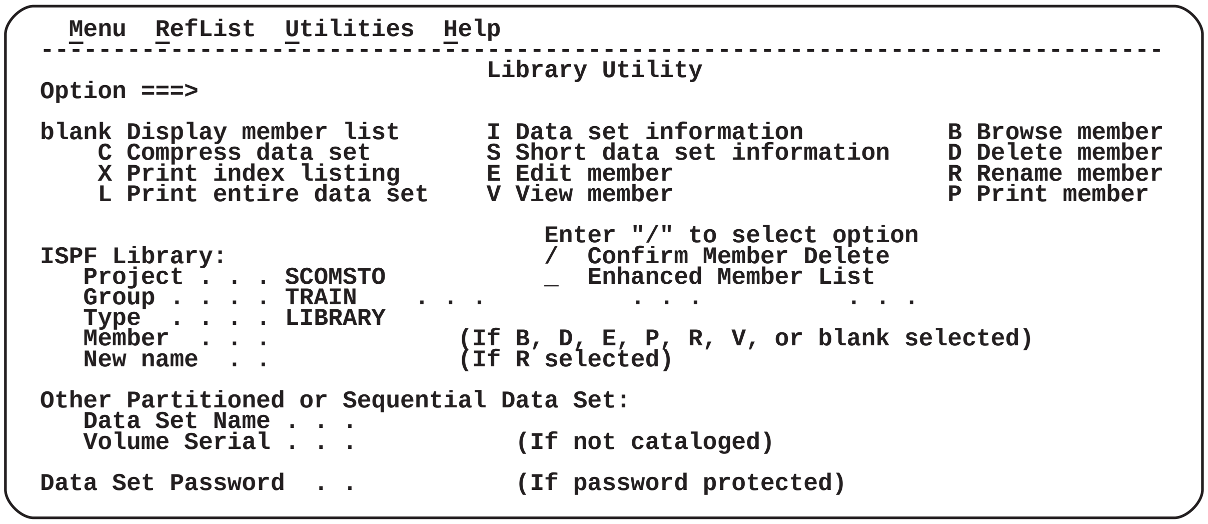
- **The rename process might allow you to rename a data set to a name not eligible for the volume the data set currently resides on**

  - **Other, similar mismatches might not be caught in time**

- **It is generally safe to change the low level qualifiers of a data set name**

The Library Utility (Option 3.1)

- ❐ If you request the Library utility, you will see a screen that looks something like this



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Library Utility
Option ===>

blank Display member list      I Data set information         B Browse member
    C Compress data set        S Short data set information   D Delete member
    X Print index listing      E Edit member                  R Rename member
    L Print entire data set    V View member                  P Print member

                                   Enter "/" to select option
ISPF Library:                      /  Confirm Member Delete
   Project . . . SCOMSTO           _  Enhanced Member List
   Group . . . . TRAIN    . . .          . . .          . . .
   Type  . . . . LIBRARY
   Member  . . .               (If B, D, E, P, R, V, or blank selected)
   New name  . .               (If R selected)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .             (If not cataloged)

Data Set Password  . .             (If password protected)
```

- ❐ Although this utility is primarily intended to work with PDSs and PDSEs, options L, I, and S can be used with sequential data sets

- ❐ Options X and L send their output to the ISPF list data set

  - ♦ Note the usage of the word “index” in Option X; the meaning here is the component we call the “directory”

Library Utility - Member List Option

❒ If you leave the option line blank, you will see a list of members in the library, perhaps something like this

```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
LIBRARY  DEPT53.PAYROL.SOURCE                               Row 00001 of 00008
Command ===>                                                  Scroll ===> CSR
   Name     Prompt          Size    Created          Changed            ID
. AARDVARK                    30   2001/12/08   2002/09/22 10:02:00    SCOMSTO
. ANTEATER                    17   2001/02/10   2001/12/08 15:01:32    SCOMSTO
. BARKER                      24   2001/02/10   2001/12/08 15:01:58    SCOMSTO
. BEAST                      258   2001/04/12   2001/12/17 14:38:11    STNT329
. CHEWER                     210   2001/04/20   2001/07/07 10:53:00    TFSHC
. GLUER                       36   2001/04/28   2001/07/13 09:11:45    SCOMSTO
. MOOVER                      38   2001/04/29   2001/07/13 09:12:04    SCOMSTO
. ZOOER                       45   2001/06/03   2002/06/04 11:49:53    STNT329
  **End**
```

❒ Once you have this list, you may

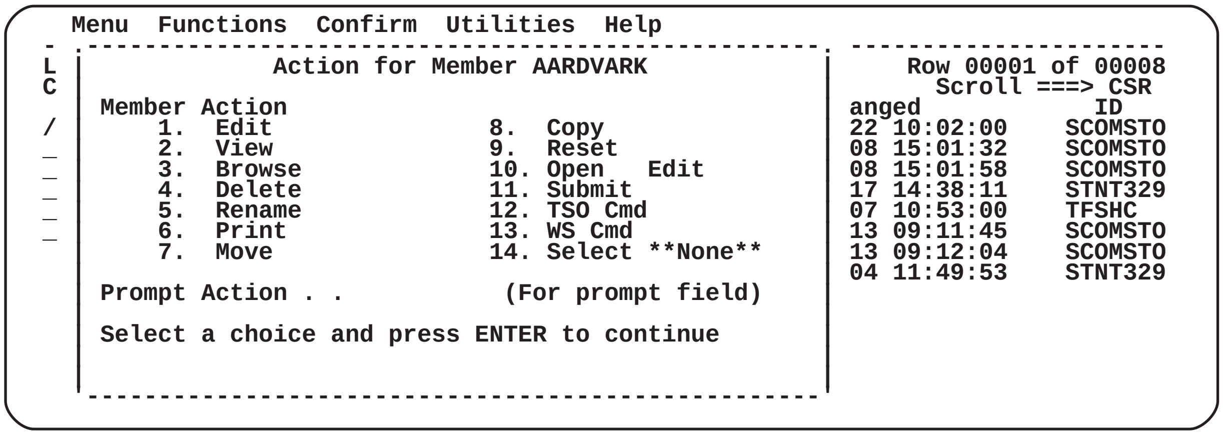
- Scroll the list up or down
- Issue a LOCATE command to find a member name
- Issue one or more line commands, next to the member name(s) you want to process, such as:

  B - Browse the member
  C - Copy the member (you are prompted for destination)
  D - Delete the member
  E - Edit the member
  G - Reset the member's statistics
  M - Move the member (you are prompted for destination)
  P - Print the member
  R - Rename the member (you must also enter the
      new name under the PROMPT column)
  T - Issue TSO command against member
  V - View the member
  W - Issue a workstation command against member

- If you selected "Enhanced Member List", the command line is 9 characters long instead of 1

Library Utility - Member List Option, continued

❒ **Alternatively, if you place a slash (/) in the line command area and press <Enter>, you will see a pop-up window that gives you a list of choices:**


```
   Menu  Functions  Confirm  Utilities  Help
 - .---------------------------------------------------. ---------------------
 L |              Action for Member AARDVARK            |     Row 00001 of 00008
 C |                                                    |       Scroll ===> CSR
   | Member Action                                      | anged            ID
 / |     1.  Edit                 8.  Copy              | 22 10:02:00    SCOMSTO
 _ |     2.  View                 9.  Reset             | 08 15:01:32    SCOMSTO
 _ |     3.  Browse               10. Open   Edit       | 08 15:01:58    SCOMSTO
 _ |     4.  Delete               11. Submit            | 17 14:38:11    STNT329
 _ |     5.  Rename               12. TSO Cmd           | 07 10:53:00    TFSHC
 _ |     6.  Print                13. WS Cmd            | 13 09:11:45    SCOMSTO
   |     7.  Move                 14. Select **None**   | 13 09:12:04    SCOMSTO
   |                                                    | 04 11:49:53    STNT329
   | Prompt Action . .            (For prompt field)    |
   |                                                    |
   | Select a choice and press ENTER to continue        |
   |                                                    |
   |                                                    |
   |----------------------------------------------------|
```

❒ **This works because the command field next to each member name is a point-and-shoot field**

- ♦ **Actually, you no longer need the slash: just place the cursor on the command line next to a member name and press <Enter> to get this menu**

Notes

- ♦ **The Prompt Action area is for:**
  - ✗ Typing in a new name for the Rename, Move, and Copy functions
  - ✗ Edit macro for Edit or View function

Library Utility - Member List Option, continued

❒ **Regarding the Action Bar choices**

- ♦ **<u>Menu</u>, <u>Utilities</u>, and <u>Help</u> provide the kind of lists of choices we've seen already**

- ♦ **<u>Functions</u> provides three choices**

  - ✗ **<u>Save List</u>** - save the member list as a file that you can view, edit, work with

  - ✗ **<u>Change Colors</u>** - change the color assignments for components of member lists using the <u>Member List Color Change Utility</u> panel

    - ➢ You can also display this panel using the **<u>MLC</u>** command from the command / option line

  - ✗ **<u>Initial Sort View</u>** ... - specify order member list should be sorted on initially

    - ➢ You can also display this panel using the **<u>MLS</u>** command from the command / option line

- ♦ **The <u>Confirm</u> action bar choice lets you require (or not) explicit confirmation when you delete a member**

❒ **We will not lecture any more on these menu items, but you might find them useful at some point in time**

Sorting Member Lists

- ❐ **Whenever you display a member list to work with, it initially comes up sorted by member name**

- ❐ **However, you can change the order of the member list by issuing a SORT command from the COMMAND line:**

  COMMAND ===> SORT [*field1* [{A|D}]] [*field2* [{A|D}]]

  - ♦ **Where either *field1* or *field2* is any of the column headers from the member list, most useful of which are**

NAME	**-**	**Member name (the default if no operands specified)**
CREATED	**-**	**Member creation date**
CHANGED	**-**	**Date and time member was last changed**
SIZE	**-**	**Number of records (lines) in the member**
ID	**-**	**TSO id of user who last updated (or created) the member**

Examples

```
SORT   CHANGED   ID
SORT   SIZE   A
SORT
```

- ❐ **NAME and ID are sorted into ascending sequence by default, all others use descending sequence**

  - ♦ **However, you may specify A (for Ascending) or D (Descending) explicitly yourself (z/OS 1.7 or later)**

- ❐ **Again, the MLS command can set the initial sort order**

Printing Data

- ❒ **We saw earlier (pp. 45-47) how ISPF allocates LIST and LOG data sets as needed**

  - ♦ **And that when you print a data set or member, the data actually goes to LIST data set**

- ❒ **We also saw how to set the default processing (which we recommended as DELETE without printing for both the LIST and LOG data sets)**

- ❒ **But suppose you really want to get hardcopy of what you've printed**

  - ♦ **You can print either the LIST or LOG data sets by issuing the LIST command or the LOG command, respectively, then requesting the output go to a printer right away**

Printing Data, continued

❐ **If you issue the command LIST, for example, you see something like this:**

```
                     Specify Disposition of List Data Set
Command ===> _

  List Data Set (SCOMSTO.SPF1.LIST) Disposition:
    Process Option . . . . 2  1. Print data set and delete
                              2. Delete data set without printing
                              3. Keep existing data set and
                                 continue with new data set
    Batch SYSOUT class . . A
    Local printer ID or
    writer-name . . . . .
    Local SYSOUT class . . _

  Press ENTER key to process the list data set.
  Enter END command to exit without processing the list data set.

Job statement information:  (Required for system printer)
 . . . //USERID   JOB  (ACCOUNT),'NAME'
 . . . //*
 . . . //*
 . . . //*
```

- ♦ **On the Process Option line, key in 1 and then specify a local printer id or at the bottom create a valid JCL JOB statement**

- ♦ **The LIST data set will be sent to the local printer or the remote printer requested**

- ♦ **You may also skip this panel by specifying the option you want in the command:**

LIST   [PRINT|KEEP|DELETE]

- ♦ **Similar remarks apply to the LOG command**

❐ **Note: from any ISPF command line, the PRINT command will send a screen image to the LIST data set**

Computer Exercise: Utility Functions

1. Use ISPF Utility to copy the data from my sequential data set, ________.TRAIN.ZINPUTX, into your sequential data set: <userid>.TRAIN.ZINPUTX

2. Use ISPF Utility to copy the data from my sequential data set, ________.TRAIN.INPUTA, into your sequential data set: <userid>.TRAIN.INPUTA

3. View <userid>.TRAIN.ZINPUTX to make sure your copy was successful (use ISPF option 1).

4. Use ISPF utility functions to accomplish these tasks:

   a. Copy the following members from ________.TRAIN.LIBRARY into your library <userid>.TRAIN.LIBRARY:

      BASE
      DATAREAS
      FILES
      INITIALZ
      INPEDTA
      INPUTC
      LOGIC
      TERMINAT

   b. Rename member INPUTC to be OLDPOEM

   c. View member INPEDTA

   d. Print your entire LIBRARY data set to your List data set

   e. Rename <userid>.TRAIN.LIBRARY to <userid>.RAIN.LIB; then name it back

   f. Delete <userid>.TRAIN.INPUTA.

5. Answer the following questions regarding ________.TRAIN.LIBRARY:

   a. Largest member ________ b. Smallest member ________

   c. Oldest member ________ d. Newest member ________

Section Preview

❒ **Productivity Tips and Techniques**

- ♦ **Quick Advance and Jump Functions**
- ♦ **Split Screen**
- ♦ **Command Stacking**
- ♦ **CMDE Command**
- ♦ **Retrieving Commands**
- ♦ **Great Tricks (Machine Exercise)**

Quick Advance and Jump Functions

❒ **When you know what option / suboption / sub...option you want to use, you don't need to step through the screens one level at a time**

<u>You can use the Quick Advance option:</u>

♦ **Code the list of option choices you want, separated by dots (.):**

```
                    ===> 3.8
      or
                    ===> 1.1.3.3
```

❒ **Similarly, when you are deep in the innards of the hierarchy of panels, you don't need to exit up the hierarchy and then step down**

<u>You can use the Jump function:</u>

♦ **Code the target option/suboption... preceded by an equals sign (=) from any line on the screen with an arrow:**

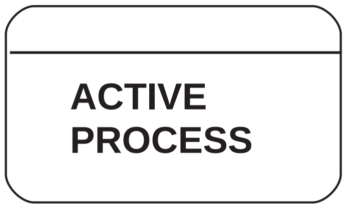
```
                    ===> =1.1.3.2
```

♦ **If you set "Jump from leader dots" on the Settings panel, you can issue jump requests from field prompts that have leader dots (. . or ...)**

Split Screen

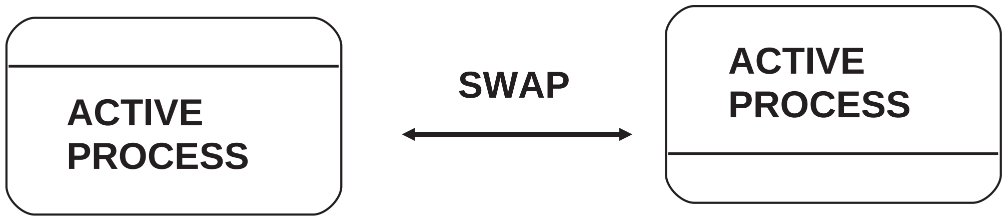
1. Move cursor to where you want the split to occur:

2. F2 splits the screen (SPLIT command); new screen becomes the active process and starts at the Primary Option Menu



3. The location of the cursor determines which is the active process

❒ Move the cursor with the arrow keys, or F9 flip-flops active screens (SWAP command)



❒ You can change the location of the split by simply moving the cursor to a new dividing point and pressing F2 again

Split Screen, continued

4. To terminate split screen ...

- ❒ **Place cursor in screen you don't want anymore**
- ❒ **Return to Primary Option Menu (F4 or RETURN command)**
- ❒ **Option X (EXIT ISPF) deletes the current screen, leaving you in the remaining screen**
  - ♦ **Or, a quick technique for doing these last two steps is ...?**
- ❒ **When you are running in GUI mode**
  - ♦ **Split causes a new physical window to be created**
  - ♦ **Only two windows are allowed, and only one may be active at a time**
  - ♦ **If you issue Split when there are already two windows, it is the same as issuing Swap: the other window becomes the active window**

Split Screen, continued

❒ **The maximum number of concurrent logical screens is 32**

- ♦ **Although IBM supplies a default maximum of 8, the product can be installed to support up to 32 separate screens**

❒ **While ISPF assigns an internal number id to each screen, you may give any logical screen a name of your choice**

- ♦ **2-8 alphanumeric characters, first is not numeric, and name cannot be one of these reserved words: NEXT, PREV, LIST, ON, OFF, PERM**

Syntax

```
SCRNAME   name   [PERM]        <-- use to assign a name
SCRNAME   {ON|OFF}             <-- set name display on panel
```

Where

- ♦ *name* **is the screen name you assign**
- ♦ **If you specify PERM, this screen name will override any screen name assigned to this screen by internal software (may only specify PERM if** *name* **is specified)**
- ♦ **ON will cause the screen name to display in the panelid area; OFF will remove the screen name (do not specify when specfying an actual** *name***)**

Split Screen, continued

❒ **Looking at the SPLIT command more closely:**

- ♦ **If there is only one logical screen, SPLIT creates a second logical screen**

- ♦ **If there are two logical screens, SPLIT changes the location of the split (on the 3270 screen), as before**

- ♦ **"SPLIT   NEW" creates a new logical screen**

❒ **Logical screens behave a little differently when there are more than two**

- ♦ **A 3270 screen can still only show two screens at a time; other logical screens may be present; they are displayed using SWAP command with options discussed on the next page**

- ♦ **When running in GUI mode, each logical screen is in its own window**

Split Screen, continued

❐ **The full syntax of the SWAP command ...**

Syntax

SWAP [LIST|PREV|NEXT|*name*|*id*]

Where

- **LIST provides a list of screens to choose from (display screen id, screen name, panel id)**
- **PREV and NEXT cycle through the screens in screen id order**
- *name* **allows you identify the chosen screen by screen name,** *id* **to identify the screen by screen number**

❐ **If you specify no parameters, the behavior is:**

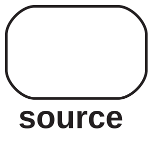
- **If there is only one logical screen: no action**
- **If there are two logical screens, swap between them**
- **If there are more than two logical screens, swaps between the two most recently displayed screens**

❐ **Choosing a logical screen, then, changes the focus (running in GUI mode) or display (running in 3270 mode)**

Split Screen, continued

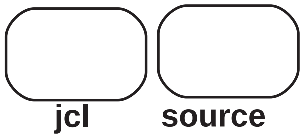
❒ **Here is an example of using multiple split screens**

- **Say you get into edit of a COBOL source program**
- **Issue the command ===>** scrname source



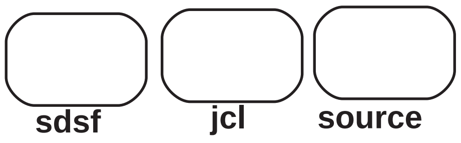
❒ **Next, split the screen**

- **Issue the command ===>** split new
- **Get into edit of some JCL to run the COBOL compiler**
- **Issue the command ===>** scrname jcl



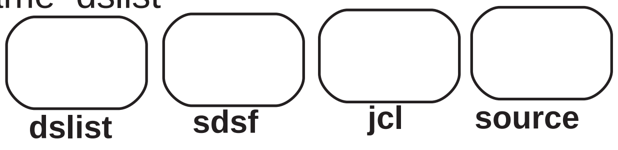
❒ **Next, split the screen again**

- **Issue the command ===>** split new
- **Get into, say, the facility for looking at the output from jobs**
- **Issue the command ===>** scrname sdsf



❒ **Next, split the screen again**

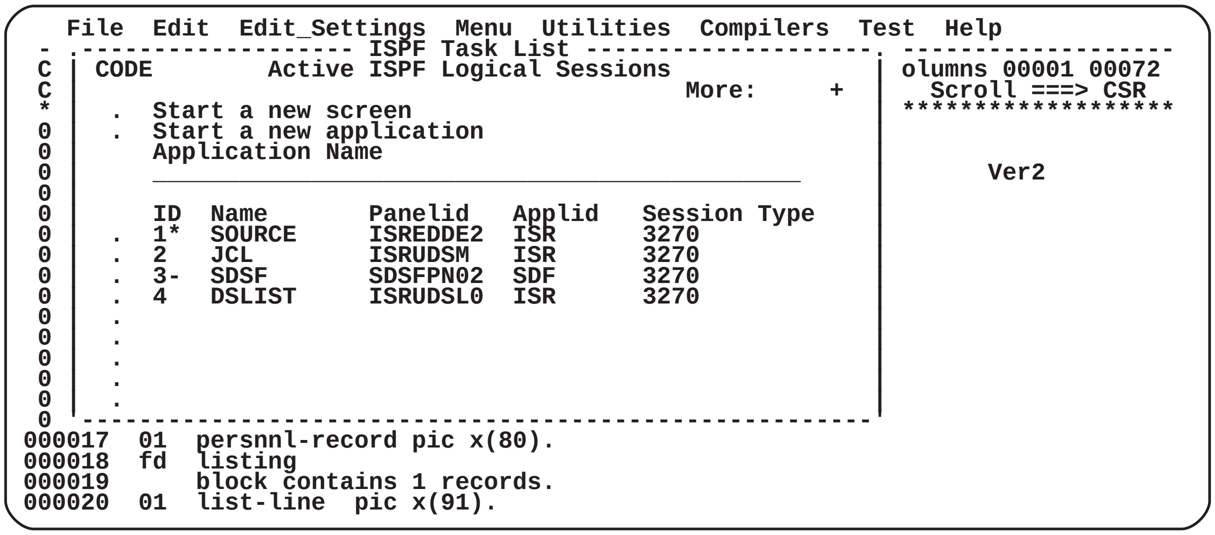
- **Issue the command ===>** split new
- **Get into, say, option 3.4 (dslist)**
- **Issue the command ===>** scrname dslist



Split Screen, continued

❒ Now, when you are in any screen, you can issue "swap   list" to see what screens are available

- ♦ You will see something like this:



```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 - .------------------- ISPF Task List --------------------. -------------------
 C | CODE        Active ISPF Logical Sessions              | olumns 00001 00072
 C |                                          More:     +  |   Scroll ===> CSR
 * |  .  Start a new screen                                | *******************
 0 |  .  Start a new application                           |
 0 |     Application Name                                  |
 0 |     ________________________________________          |      Ver2
 0 |                                                       |
 0 |     ID  Name        Panelid   Applid   Session Type   |
 0 |  .  1*  SOURCE      ISREDDE2  ISR      3270           |
 0 |  .  2   JCL         ISRUDSM   ISR      3270           |
 0 |  .  3-  SDSF        SDSFPN02  SDF      3270           |
 0 |  .  4   DSLIST      ISRUDSL0  ISR      3270           |
 0 |  .                                                    |
 0 |  .                                                    |
 0 |  .                                                    |
 0 |  .                                                    |
 0 |  .                                                    |
 0 '-------------------------------------------------------'
000017  01  persnnl-record pic x(80).
000018  fd  listing
000019      block contains 1 records.
000020  01  list-line  pic x(91).
```

- ♦ And now you can get to a series of screens quickly and effectively

❒ Also, you can issue "swap   2" or any other screen number

❒ Think about having "split new" and "swap list" assigned to function keys

❒ Note also that you can start another new screen from here

- ♦ You can even start a whole new application, but that is not discussed in this class

❒ In z/OS V1.10, you can issue the command "SWAPBAR" to have a line of point-and-shoot fields, one for each split screen running

Command Stacking

❐ You can issue a series of commands on any command line, each command separated from the previous by a semicolon (;)

❐ The commands will be executed in sequence (unless a syntax or logical error is found)

❐ Suppose you are running view in each of two screens, as you are comparing two files to ensure they are the same

❐ You could step through both files by issuing this series of commands to scroll down both files:

```
===> DOWN 6;SWAP;DOWN 6;SWAP
```

❐ You may issue any command that is valid for the screen that is current when the command is issued

- ♦ Commands in a stack following HELP or RETRIEVE commands are deleted

❐ You won't see the intermediate steps, just the final display

❐ Such a stack of commands may be assigned to a function key

❐ You can change the command stacking separator character from the ';' to almost any other character, using ISPF Settings

Command Stacking, 2

❒ Stacking commands assumes you can remember the commands available

♦ Here is a place to summarize the ISPF commands we have mentioned so far:

❒ ISPF General Commands

❒ Member List Commands

❒ Edit / View Commands

❒ Scrolling / Positioning Commands

❒ Tutorial Commands

Command Stacking, 3

❐ **Also note that coding two consecutive command delimiters simulates pressing <Enter>**

- ♦ **For example, suppose you are not currently in split screen mode; then**

```
===>  split;2;;l exer
```

- ♦ **Will cause the following sequence of events**

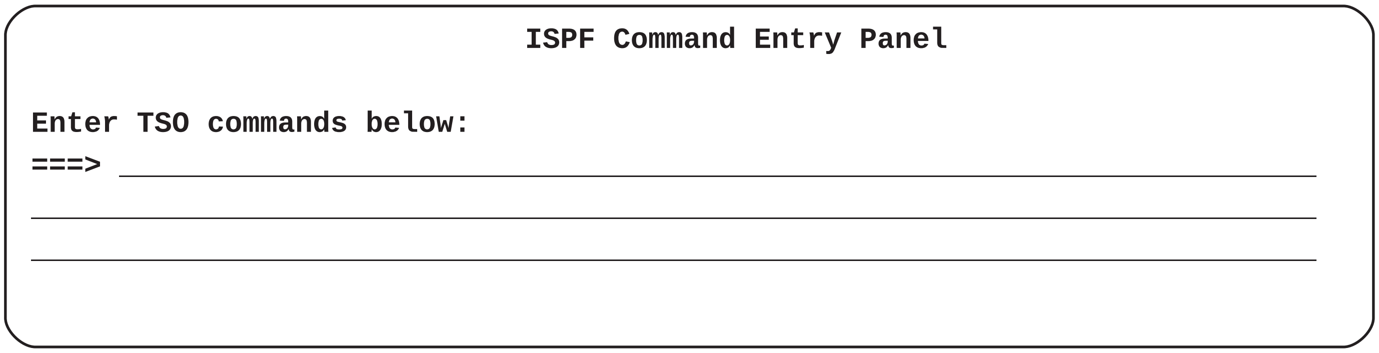
  - ✗ The screen will be split
  - ✗ Control will go to the Edit entry panel
  - ✗ Enter is simulated; if you had a library name in the Edit entry panel from earlier, control goes to the member list for that library
  - ✗ The member list is positioned to the member with name exer
    - ➢ Or to where such a member would be, if there is not currently one in the list

CMDE Command

❒ **Sometimes you need more room than there is available on the command line**

❒ **You can issue the CMDE ISPF command**

- ♦ **This command provides a panel with just a command line on it:**



```
                         ISPF Command Entry Panel

Enter TSO commands below:
===> ____________________________________________________________________
_________________________________________________________________________
_________________________________________________________________________
```

- ♦ **You may enter up to 234 characters of command on this line, including stacked commands**

  ✗ Even though the panel says "Enter TSO commands", you may also enter ISPF commands

Retrieving Commands

❒ **Often you want to issue a long command again, and it would be nice not to have to re-key the command in**

For example

```
Command ===> f   'Now is the time'c  last 22 59
```

- ♦ **Press <Enter> and the find command is executed**
- ♦ **Now, to re-issue the command, you could, of course just press F5 (RFIND)**
- ♦ **But, if you want to make changes, you can use the Retrieve command to pull up the whole text of the command, then make any changes in the command, then press <Enter>**
  - ✗ This is most easily accomplished if Retrieve is assigned to a Function key

❒ **Similarly, it is often nice to re-issue a command that you used several commands ago**

❒ **ISPF maintains a stack of all commands issued from the command / option line, and it provides three Retrieve commands that can pull commands off this stack**

- ♦ **The retrieve stack is 255 bytes long (shared by both screens if you are in split screen mode)**
- ♦ **The number of commands that can be saved on the retrieve stack varies with the length of the commands issued; fractions of a command string are not saved**

Retrieving Commands, 2

- ❐ **Historically, ISPF used F12 for the Retrieve command**

  - ♦ **However, this is not CUA-compliant: F12 is generally assigned to the Cancel command**

  - ♦ **Consider using F24 for Retrieve (we will assume this in our examples)**

- ❐ **When you press F24, then, ISPF re-displays the most recently issued command**

  - ♦ **Each subsequent time you press F24, the previous command in the retrieve stack is displayed**

  - ♦ **If you cycle throught the entire stack, the command at the top of the retrieve stack is re-re-displayed, and so on**

- ❐ **Sometimes you may go too fast and bypass the command you were after**

  - ♦ **Rather than re-cycling through the entire retrieve stack, you can issue the <u>RETF</u> command (for Retrieve Forward) ...**

Retrieving Commands, 3

- **If you issue an RETF command after a Retrieve command, you cycle through the retrieve stack from the last displayed command in a forward direction: progressing to more recent commands**

- **If you issue RETF after any command except Retreive (or RETF), you process the entire retrieve stack starting with the oldest, cycling through the most recent**
  - ♦ **Again, it helps to assign RETF to a Function key**

- **Another alternative is to issue the <u>RETP</u> command: present a pop-up window containing the last 25 commands in the retrieve stack**
  - ♦ **You can then select one and it will be copied to the command / option line for you to work with**

- **If you retreive a long command (one entered, say, using CMDE), the command is truncated**
  - ♦ **However, you can issue CMDE and then issue a RETF or RETP command, and then retrieve the command into the CMDE window**

Tips and Techniques Summary

❒ **Here is a summary of the shortcuts, tips, and techniques available that we have discussed**

Quick Advance

- ♦ **From an option line, move forward by entering option.sub_option[.sub_sub_option ...]**

Jump

- ♦ **From any input line, jump to any panel by entering =option[.sub_option[.sub_sub_option ...]]**

Split Screen

- ♦ **Split command (F2) gives you multiple activities available**
- ♦ **Swap command (F9) swaps among activities**
- ♦ **In 3270 mode, with two windows, change the proportions of the windows by moving the cursor and re-issuing Split (press F2 again)**
- ♦ **End a split screen by jumping out from it (=X in any input field)**
- ♦ **Assign a name to a screen using Scrname command**
- ♦ **In z/OS 1.10 and later, issue Swapbar command to get line at bottom of panel for navigating**

Tips and Techniques Summary, 2

Command stacking

- **From a command / option line, enter a series of commands separated by semi-colons**

CMDE command

- **To get a long command line**

Command retrieval

- **RETRIEVE, RETF, RETP - to save a lot of typing**

❒ **Identifiers - as of z/OS 1.7, you may request multiple identifier values to be displayed on your panels**

- **Up to 17 characters will appear on the left hand side of the panel title line, as many as will fit**
- **The identifiers are displayed or not based on commands (in order of priority of display):**
  - ✗ SYSNAME {ON|OFF}
  - ✗ USERID {ON|OFF}
  - ✗ PANELID {ON|OFF}
  - ✗ SCRNAME {ON|OFF}

<u>Computer Exercise: Great Tricks</u>

1. Logon to TSO and get into ISPF.

2. Get into View of member INPEDTA in <userid>.TRAIN.LIBRARY; position yourself at line 177.

3. Split the screen; we refer to your View-session as your <u>first screen</u>, and your new session as your <u>second screen</u>; in your second screen, quick advance to the Library utility (3.1)

   a. Assign the Retrieve command to some function key

   b. Get a member list of ________.TRAIN.LIBRARY

   c. View member INPEDTA; position yourself at line 177

   d. Re-split the screen at about the half-way point, and verify both windows contain the same data

   e. Stack the commands <u>Down 8;swap;Down 8</u> on either command line, and press Enter; watch what happens

   f. Press the key assigned to the Retrieve command, then press <Enter>

   g. In your second screen, END the view, END the member list, and jump to the move/copy utility (=3.3)

   h. Copy member INPUT1 from ________.TRAIN.LIBRARY to your <userid>.TRAIN.LIBRARY, changing the member's name to ARAGON in the process

   i. End split screen mode by jumping out of your second screen.

4. You are now in View of <userid>.TRAIN.LIBRARY(INPEDTA)

5. Issue the RETP command; examine the contents of the pop-up window; select a command and ask ISPF to execute it.

6. Logoff of TSO.

Section Preview

❐ **EDIT**

- ♦ **EDIT**
- ♦ **Sequence numbers**
- ♦ **Nulls**
- ♦ **Line commands**
- ♦ **Working with EDIT (Machine Exercise)**

EDIT

❒ **Works with sequential data sets, members of PDSs, or data on the workstation (not discussed here)**

- ✗ **Enter new data**
- ✗ **Modify existing data**

❒ **Requires update access**

- ✗ **Exclusive use of data set or member**
- ✗ **Update access authority (data security package)**

❒ **Remember, Edit and View are the same except**

- ♦ **Edit can change data and then save the changes in place**
  - ✗ In the member or sequential data set
- ♦ **View cannot save changes in place**

❒ **So everything we say about Edit on the following pages also applies to View, except where explicitly noted**

Sequence Numbers

- ❒ **Data being edited may have the attribute of numbers ON or OFF**
  - ♦ **ON means that sequence numbers are stored in the data records**
  - ♦ **OFF means the data is unnumbered**

- ❒ **Even for unnumbered data, Edit displays sequence numbers on the screen for ease of reference; the numbers simply aren't stored in the data**

- ❒ **For data with the "NUMBER ON" attribute, sequence numbers may be:**
  - ♦ **STANDARD (last 8 columns for fixed length records, first eight columns for variable length records)**
  - ♦ **COBOL (columns 1-6, only fixed length records supported)**
  - ♦ **BOTH (if fixed length records)**

- ❒ **During editing, sequence numbers are always displayed on the left of the screen**

- ❒ **For STANDARD numbers, if STATISTICS option is ON, the first six digits are the sequence number, the last two are a level number within the current version**

Creating Data Using EDIT

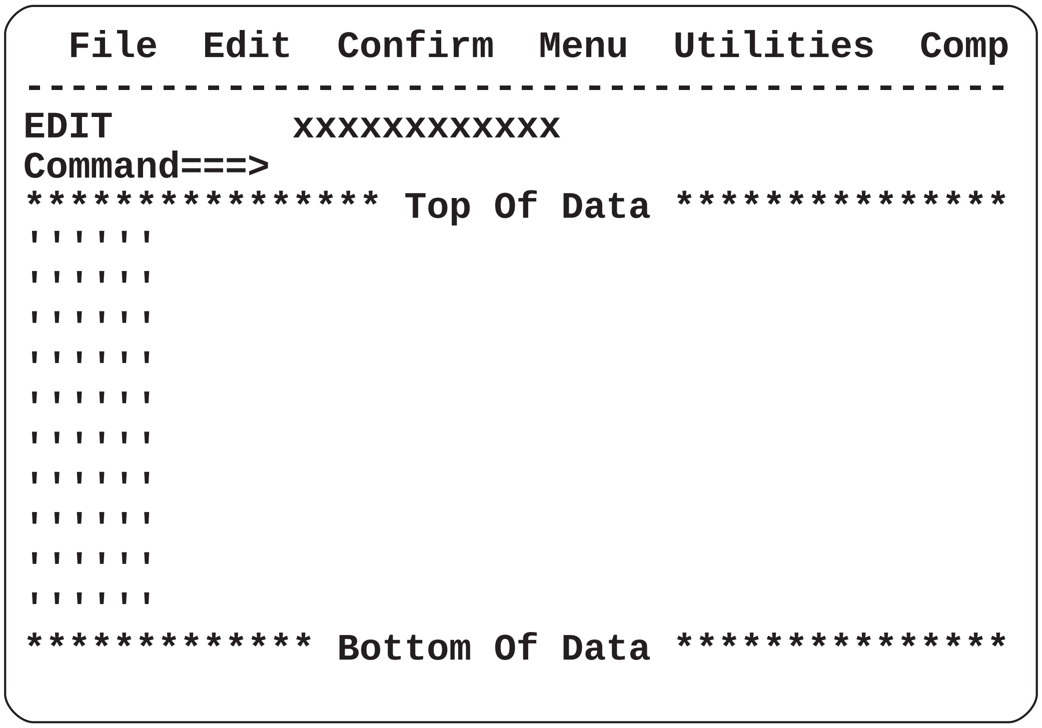
❒ **Specify member or sequential data set to Edit**

- **If the sequential data set exists (has been allocated) but is empty, a panel of empty lines is provided for entering new data**

- **If a specified member does not exist, but the library does, a panel of empty lines is provided for entering new data**

  - ✗ An "empty line" is a line where the sequence numbers show as apostrophes (' ' ' ' ' ') and there is no data on the line

  - ✗ A "screen of empty lines" looks something like this:



```
  File  Edit  Confirm  Menu  Utilities  Comp
--------------------------------------------
EDIT        xxxxxxxxxxxx
Command===>
**************** Top Of Data ***************
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
************* Bottom Of Data ***************
```

Entering New Data

- Initially, the cursor is positioned at the command line

- Use the NEW-LINE key or TAB key to get to the first data line

- TAB past the sequence number field (which is initially filled with apostrophes) to begin typing data

- Use the NEW-LINE or TAB keys for each new line

```
EDIT       xxxxxxxx
Command===>
*************** Top Of Da
'''''' THIS IS LINE 1
'''''' THIS IS NOT LINE 1
'''''' THIS SNOT IS LIKE
'''''' THE ISN'T TIME TO
'''''' THIS SNOT ISN'T A
'''''' WHAT'S NOT THE _
```

Entering New Data, continued

❒ **When done, or screen full, press the <Enter> key**

- ♦ **If the screen is full, each line being non-empty:**

  - ✗ **All lines from the cursor up will have sequence numbers assigned, the data will be scrolled up one line, and the cursor will be positioned to the last line**

```
EDIT        xxxxxxxx
Command===>
*************** Top Of Da
'''''' THIS IS LINE 1
'''''' THIS IS NOT LINE 1
'''''' THIS SNOT IS LIKE
'''''' THE ISN'T TIME TO
'''''' THIS SNOT ISN'T A
'''''' WHAT'S NOT THE _
```

```
EDIT        xxxxxxxx
Command===>
000100 THIS IS LINE 1
000200 THIS IS NOT LINE 1
000300 THIS SNOT IS LIKE
000400 THE ISN'T TIME TO
000500 THIS SNOT ISN'T A
000600 WHAT'S NOT THE
'''''' _
```

(Press ENTER)

  - ✗ **From now on, you can enter one line at a time, and each time you press <Enter>, the data will continue to scroll up one line**

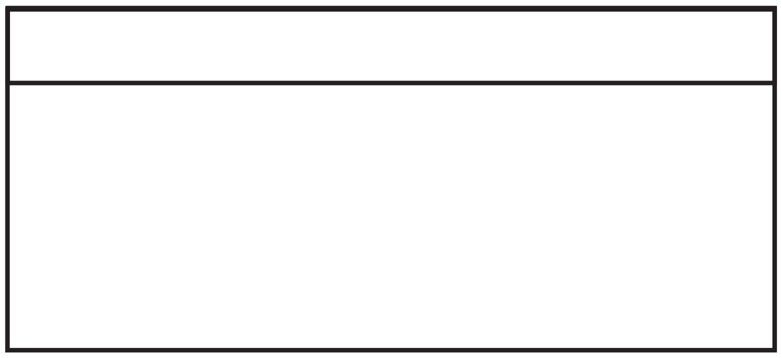
  - ✗ **Until you press <Enter> with at least one empty line left on the screen, then the last line will be closed up**

❒ **If there are still empty lines on the screen when you press <Enter>, sequence numbers are assigned to lines with data on them and the empty lines are deleted**

Sequential and Partitioned Data Sets

❒ **When you allocate a new sequential disk data set, you create a space for placing records:**

❒ **When you allocate a PDS, you create a directory with space for member names and space for holding members that will contain records:**



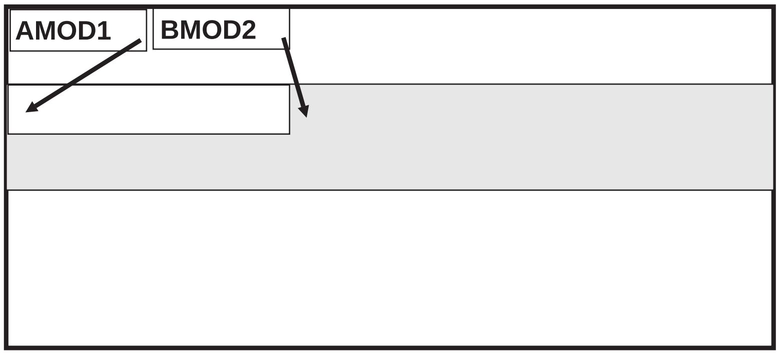
❒ **Using ISPF, you place records in these spaces by Copy, Move, or Edit**

Using Sequential and Partitioned Data Sets

❐ **Before you can Edit, Copy to, or Move to new space on disk you must have first allocated the space:**

- **Well, you can allocate the destination file at the time you Copy, or Move, but you must still pre-allocate a sequential file to edit in it**

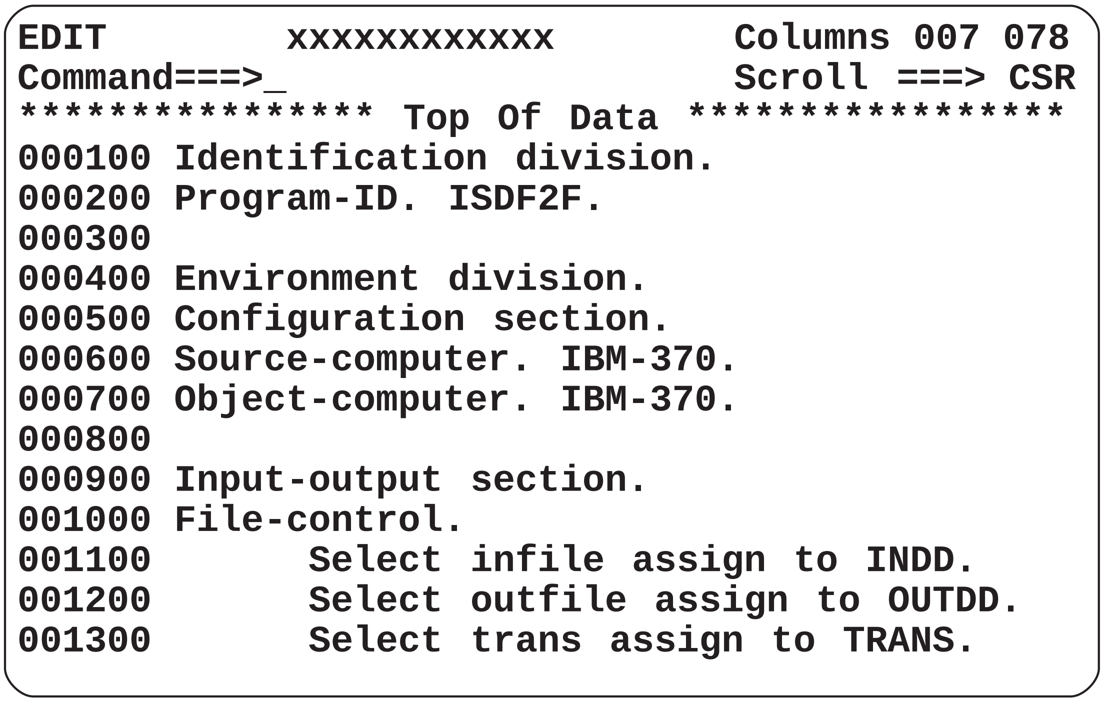
❐ **To create a new member in a PDS, it is sufficient that the PDS has been allocated; it may or may not have any other members in it at the time you try to create a new member:**



Editing Existing Data

❒ **Specify member or sequential data set as before**

❒ **If the sequential data set or member exists and is not empty, the first screen of data is displayed:**



```
EDIT        xxxxxxxxxxxx       Columns 007 078
Command===>_                   Scroll ===> CSR
**************** Top Of Data *****************
000100 Identification division.
000200 Program-ID. ISDF2F.
000300
000400 Environment division.
000500 Configuration section.
000600 Source-computer. IBM-370.
000700 Object-computer. IBM-370.
000800
000900 Input-output section.
001000 File-control.
001100       Select infile assign to INDD.
001200       Select outfile assign to OUTDD.
001300       Select trans assign to TRANS.
```

❒ **If the sequential data set or partitioned data set does not exist (has not had space allocated), ISPF issues an error message**

Typeover, Insert Key, Delete Key, Erase EOF Key

❒ **When you are editing existing data, the simplest changes can be made by simply moving the cursor to the appropriate spot and:**

- ♦ **Typing over the data to change**

- ♦ **Use the insert key to insert data where there's room on a line**

- ♦ **Use the delete key to delete individual characters and have the rest of the line close up to the left**

❒ **The End key (on PC emulators; the Erase EOF key on mainframe terminals) is useful for clearing the rest of a line after the cursor location**

- ♦ **Sometimes on PCs, the End key is set to move the cursor to the end of the line**

  - ✗ The setting is usually changeable from the emulator

Nulls

❐ **When you are inserting data on a line, if you press the insert key and begin typing, you may find an error figure on the lower left corner of your screen, and the keyboard may lock up**

- ♦ **The problem is that the line is full, so you can't insert any more characters!**

- ♦ **Even though it may look like there is space, the line is filled with blanks, and they take up room like other characters:**

```
023000 MOVE HOLD-DATE TO TRANS-DATE.~~bbbbbbbbbbbbbbbbb~~
```

❐ **First, press the <u>RESET key</u> to clear the error figure and unlock the keyboard**

- ♦ **This also takes you out of insert mode**

Nulls, continued

❒ **Now, there are two ways to solve this problem:**

First, step by step:

- **Position the cursor one character past the last non-blank character of the line on which you want to insert characters:**

023000 MOVE HOLD-DATE TO TRANS-DATE.~~bbbbbbbbbbbbbbbbbb~~

- **Press the Erase EOF key (or End key):**

```
023000 MOVE HOLD-DATE TO TRANS-DATE._
```

- **Position the cursor back to the point of insertion:**

```
023000 MOVE_HOLD-DATE TO TRANS-DATE.
```

- **Press the insert key and type in the data you want to insert:**

```
023000 MOVE CORRESPONDING_HOLD-DATE TO TRANS-DATE.
```

Nulls, 3

Alternatively

♦ Get the cursor to the command line and issue this command:

```
Command ===> nulls on
```

✗ This will replace trailing blanks with trailing null characters in all lines:

Before

```
023000 MOVE HOLD-DATE TO TRANS-DATE.~~bbbbbbbbbbbbbbbbbb~~
```

After

```
023000 MOVE HOLD-DATE TO TRANS-DATE.
```

✗ You can now insert characters in any line with trailing blanks, once you press the insert key

❒ Pressing ENTER or RESET or a function key takes you out of insert mode

❒ When your file is saved, nulls are automatically replaced with blanks

❒ Edit will remember that nulls are on (even across TSO sessions) until you issue

```
Command ===> nulls off
```

EDIT Line Commands

COLS	-	show column numbers			
I	-	Insert line(s):	I	I*n*	
D	-	Delete line(s):	D	D*n*	DD - - - DD
R	-	Repeat line(s):	R	R*n*	RR - - - RR
			RR - - - RR*n*		
X	-	Exclude line(s) from display:	X	X*n*	XX - - - XX
S	-	Show excluded line(s) with fewest leading blanks	S	S*n*	
F	-	Show First excluded line(s):	F	F*n*	
L	-	Show Last excluded line(s):	L	L*n*	
M	-	Move line(s):	M	M*n*	MM - - - MM
C	-	Copy line(s):	C	C*n*	CC - - - CC
A	-	After this line (target for M or C):	A	A*n*	
B	-	Before this line (target for M or C):	B	B*n*	

✗ <u>M</u>ove and <u>C</u>opy require A or B to complete

EDIT Line Manipulation - Insert

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
I50600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION  COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
''''''          _
''''''
''''''
''''''
''''''
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
```

EDIT Line Manipulation - Delete

```
  000100 PGM1     CSECT
  000200          SAVE   (14,12)
  000300          USING  PGM1,12
  000400          LR     12,15
  000500          ST     13,SAVE+4
  000600          LR     2,3
  000700          LA     13,SAVE
  000800          ST     13,8(2)
  000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
→ D01000          BAL    3,GETDATE
  001100          BAL    3,SETPARMS
→ DD1200          BAL    3,ATTCHSUB
  001300          WTO    'INITIALIZATION COMPLETE'
  001400          BAL    3,READTRAN
→ DD1500          OI     SW,1
  001600 BIGLOOP  EQU    *
  001700          TM     SW,1
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001100          BAL    3,SETPARMS
001600 BIGLOOP  EQU    *
001700          TM     SW,1
001800          BO     EXLOOP1
001900          L      5,TRKEY
002000          CVD    5,DBLWRD
002100          SP     DBLWRD,RNDFACT
002200          MVO    DBLWRD+5(3),DBLWRD+6(2)
```

EDIT Line Manipulation - Repeat

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
R00400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
RR1000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
00RR00          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000410          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
001210          BAL    3,GETDATE
001220          BAL    3,SETPARMS
001230          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
```

EDIT Line Manipulation - Move

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
B00900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
M01100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000810          BAL    3,SETPARMS
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001200          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

EDIT Line Manipulation - Copy

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
CC0500          ST     13,SAVE+4
000600          LR     2,3
CC0700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
A21000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
000800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001010          ST     13,SAVE+4
001020          LR     2,3
001030          LA     13,SAVE
001040          ST     13,SAVE+4
001050          LR     2,3
001060          LA     13,SAVE
001100          BAL    3,SETPARMS
```

EDIT Line Manipulation - eXclude

```
000100 PGM1      CSECT
000200           SAVE   (14,12)
X00300           USING  PGM1,12
000400           LR     12,15
000500           ST     13,SAVE+4
000600           LR     2,3
XX0700           LA     13,SAVE
000800           ST     13,8(2)
000900           OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000           BAL    3,GETDATE
00XX00           BAL    3,SETPARMS
001200           BAL    3,ATTCHSUB
001300           WTO    'INITIALIZATION COMPLETE'
001400           BAL    3,READTRAN
```

```
000100 PGM1      CSECT
000200           SAVE   (14,12)
- - - - - - - - - - - - - - 1 Line(s) not Displayed
000400           LR     12,15
000500           ST     13,SAVE+4
000600           LR     2,3
- - - - - - - - - - - - - - 5 Line(s) not Displayed
001200           BAL    3,ATTCHSUB
001300           WTO    'INITIALIZATION COMPLETE'
001400           BAL    3,READTRAN
001500           OI     SW,1
001600 BIGLOOP   EQU    *
001700           TM     SW,1
001800           BO     EXLOOP1
```

- ♦ **Use F or F*n* to un-exclude the First n lines that are hidden**
- ♦ **Use L or L*n* to un-exclude the Last *n* hidden lines**
- ♦ **Use S or S*n* to Show the *n* hidden lines with the fewest leading blanks**
- ♦ **Use primary command RESET to un-exclude all hidden lines**

EDIT - Line Commands To Shift Data

<	-	Data shift left:	<	<*n*	<< - - - <<	<< - - - <<*n*
>	-	Data shift right:	>	>*n*	>> - - - >>	>> - - - >>*n*
(	-	Column shift left:	(	(*n*	((- - - ((	((- - - ((*n*
)	-	Column shift right:	)	)*n*	)) - - -))	)) - - -))*n*

❒ ***n*** **= number of columns to shift (default is 2)**

Data shift:

- **do not shift non-blank data off line**
- **other restrictions, based on general structure of programming languages**

Column shift:

- **allow non-blank data to shift off line (data is lost)**
- **shift regardless of content of line**

EDIT - Line Commands To Change Case

LC	-	Force to lower case:
		LC LC*n* LCLC - - - LCLC LCC - - - LCC
UC	-	Force to upper case:
		UC UC*n* UCUC - - - UCUC UCC - - - UCC

Note:

- **these commands do not change the mode of data entry you are in (CAPS ON or CAPS OFF)**

Some More Points on EDIT

- ❐ **When lines are excluded, the message lines (*e.g.*: "5 Line(s) not Displayed") are sometimes called <u>shadow lines</u>**
  - ♦ **These shadow lines can be removed from the display using the HIDE primary command (beginning in z/OS 1.6; discussed later)**

- ❐ **Edit and view can work with records (lines) up to 32,760 bytes long**

- ❐ **The scrolling commands (and their corresponding function keys) work as before:**

UP	**(F7/F19)**
DOWN	**(F8/F20)**
LEFT	**(F10/F22)**
RIGHT	**(F11/F23)**

- ❐ **Leave Edit with the END command (F3/F15) or the RETURN command (F4/F16)**
  - ♦ **This saves the data as it currently exists on your screen (and in memory) back on disk, replacing any previous version**
    - ✗ Except that messages from the editor, column lines, and sequence numbers for unnumbered files are not saved

<u>For View, any changes are ignored on exit (End or Return)</u>

Computer Exercise: Working With EDIT

1. In your training library, create a new member called PDFDATA1. Enter the following lines of data:

```
LIBERTY FLBJIBIT IS FLELLING AWONG
FARDER DEE ORDRUNG NOW KEELINGS BEMONG
HOWELL WELLOW, KEAR FLUBBER, IST BUMMING THE LORKS?
NEVER DEE EVER, HIME LIBBER, BELLOW TWO GORKS.
```

2. On the command line, enter the command NUM OFF. This ensures the data is unnumbered, so that the displayed sequence numbers are recalculated after each operation that adds, deletes, or re-arranges lines.

3. Copy the first two lines after the fourth line.

4. Insert the following two lines after the fifth line:

```
HABER DIE BABER DEE OLBER THE LELL
ALGREN LIST ALGREN BUNTER THEE PELL
```

5. Repeat lines 3 through 8, 5 times.

6. Delete lines 17 through 24.

7. On line 2, insert the word LARDLY just before BEMONG.

8. On line 5, change FLELLING to AXLING.

9. Shift the data on lines 10 through 18 right 6 positions (using column shift, not data shift).

10. Force lines 9, 11, and 20 to be lower case (this may not be apparent, depending on your terminal and emulator program).

Section Preview

❐ **Edit and View Primary Commands**

- ♦ **COLS**
- ♦ **EXCLUDE, FLIP, HIDE**
- ♦ **LOCATE**
- ♦ **FIND, CHANGE**
- ♦ **RFIND, RCHANGE**
- ♦ **DELETE**
- ♦ **SAVE, CANCEL**
- ♦ **HEX**
- ♦ **RESET**
- ♦ **UNDO**
- ♦ **More work with EDIT (Machine Exercise)**

Edit and View Primary Commands

❒ Primary commands are entered on the comand line of the panel, as opposed to line commands that are entered in the sequence number field of the data

❒ We start with this list ...

COLS	—	put non-scrollable column number line at top of data area
EXCLUDE	—	exclude lines from the display, leaving shadow lines to point out where lines have been excluded
HIDE	—	remove shadow lines from display
FLIP	—	reverse the exclude status of all lines
LOCATE	—	specify line number; already discussed
FIND	—	specify character string or picture; reviewed here
RFIND	—	repeat last FIND command; reviewed here
CHANGE	—	specify old string and new string
RCHANGE	—	repeat last CHANGE command
DELETE	—	delete lines
SAVE	—	preserve current changes, continue to edit current data; no operands; changes not actually saved under View; not discussed further
CANCEL	—	restore to status since last SAVE; no operands; not discussed further
HEX	—	display data in hexadecimal or character format; discussed earlier
RESET	—	delete message lines, COLS lines, *etc.*; restore excluded lines; not discussed further
UNDO	—	backout a previous modification of the data being edited / viewed

COLS - Edit / View Primary Command

```
COL[S]
```

- **Puts a non-scrollable column number of line at the top of the data area**

- **May code as COL or COLS**

- **Removed by re-issuing COL[S] - it's a toggle command**

- **Not the same as COLS line command (which inserts a scrollable column number line)**

  - ✗ It is the same as the COLS primary command in Browse mode, discussed earlier

- **Introduced in z/OS 1.6**

EXCLUDE - Edit / View Primary Command

```
EXCLUDE   string   [NEXT]     [CHARS]    [col-1]   [col-2]
                   [ALL]      [PREFIX]
                   [FIRST]    [SUFFIX]
                   [LAST]     [WORD]
                   [PREV]
```

string	—	**string whose appearance causes line(s) to be excluded**
NEXT	—	**begin at the cursor location and scan forward**
ALL	—	**exclude all lines in the file with an occurrence of** *string*
FIRST	—	**go to top, search forward, exclude the first line containing** *string*
LAST	—	**go to bottom, search up, exclude the last line containing** *string*
PREV	—	**begin at cursor location, scan backward (up)**
CHARS	—	**locate** *string* **as standalone or embedded in words**
PREFIX	—	**only look for** *string* **at the start of a word**
SUFFIX	—	**only look for** *string* **at the end of a word**
WORD	—	**only look for** *string* **as a standalone word**
col-1	—	**left column boundary to restrict search**
col-2	—	**right column boundary to restrict search**

FIND - Edit / View Primary Command

```
FIND   string   [NEXT]    [CHARS]    [X]    [col-1]   [col-2]
                [ALL]     [PREFIX]   [NX]
                [FIRST]   [SUFFIX]
                [LAST]    [WORD]
                [PREV]
```

string	—	**string to search for**
NEXT	—	**begin at cursor location and scan forward**
ALL	—	**find all occurrences in file (position cursor at first)**
FIRST	—	**go to top, search forward for first occurrence of** *string*
LAST	—	**go to bottom, search up to find the last occurrence**
PREV	—	**begin at cursor location, scan backward (up)**
CHARS	—	**locate** *string* **as standalone or embedded in words**
PREFIX	—	**only look for** *string* **at the start of a word**
SUFFIX	—	**only look for** *string* **at the end of a word**
WORD	—	**only look for** *string* **as a standalone word**
X	—	**only search in lines that have been excluded**
NX	—	**only search in lines that have not been excluded**
col-1	—	**left column boundary to restrict search**
col-2	—	**right column boundary to restrict search**

Strings - Review

❐ Remember, the FIND, CHANGE, and EXCLUDE Edit and View commands all reference "strings"

❐ You may use any one of five kinds of strings:

Simple string

- A string not starting or ending with an apostrophe or quotation mark, and without embedded blanks, commas, or asterisks

Delimited string

- Any string bounded by apostrophes but not containing any embedded apostrophes, or bounded by quotes but not containing any embedded quotes

Character string

- A delimited string preceded or followed by a C; case sensitive

Hexadecimal string

- Any delimited string of valid hex digits preceded or followed by a character X

Picture string

- Any delimited string of picture characters preceded or followed by the letter P

Strings - Review, 2

Picture strings

- ♦ Identify types of characters in positions

Picture character	Meaning
=	any character
¬	any non-blank character
.	any non-displayable character
#	any numeric character, 0 - 9
-	any non-numeric character
@	any alphabetic character, any case
<	any lowercase alphabetic character, a - z
>	any uppercase alphabetic character, A - Z
$	any special character (non-alphanumeric)

- ♦ Picture strings can include alphanumeric characters, which represent themselves [in a non-case-sensitive way]

❐ Recall, also, that using an unquoted asterisk for a FIND string says to use the last used FIND string

EXCLUDE / FIND - Usage Technique

❒ **If you want to see all lines containing a particular character string, try this:**

- ♦ **Exclude all lines from the display (use line or primary exclude)**
- ♦ **Issue a Find command for the string you want to see**
- ♦ **The lines containing the string will "pop up" on the screen**

<u>For example</u>

- ♦ **You could request to see all lines containing the string MASTER-FILE by typing in this on the command line:**

```
Command   ===>   x  all;f  master-file  all
```

HIDE - Remove Shadow Lines

❒ **When you exclude lines from the display, shadow lines are inserted as a visual reminder as to where lines have been excluded**

- **for example ...**

shadow lines

```
000392               set part-index to 1
000393               initialize parts-table
000394               perform until more-records = 'N'
- - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 3 Line(s) not Displayed
000398                         parts-on-hand(part-index)
- - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 1 Line(s) not Displayed
000400                         parts-on-ord(part-index)
- - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 2 Line(s) not Displayed
000403                  perform read-a-record
000404                  if more-records = 'Y'
```

❒ **The HIDE command removes the shadow lines, freeing up more screen "real estate" so you can see more at one time**

Syntax

HIDE {EXCLUDED | EXCLUDE | EXC | EX | X}

- **Must specify one of the operands; they all have the same meaning**

- **Introduced in z/OS 1.6**

FLIP - Reverse EXCLUDE Status

❐ **Another technique that is sometimes useful is to un-exclude all excluded lines and exclude all non-excluded lines; the FLIP command accomplishes this**

<u>Syntax</u>

```
FLIP
```

CHANGE - Edit / View Primary Command

```
CHANGE   string-1   string-2   [NEXT]    [CHARS]   [X]   [col-1]   [col-2]
                               [ALL]     [PREFIX]  [NX]
                               [FIRST]   [SUFFIX]
                               [LAST]    [WORD]
                               [PREV]
```

string-1	—	**current string value or picture**
string-2	—	**new string value or picture**
NEXT	—	**begin at cursor location, scan forward**
ALL	—	**change all occurrences in the file (position cursor at first)**
FIRST	—	**go to top, search forward, change the first occurrence**
LAST	—	**go to bottom, search up, change the last occurrence**
PREV	—	**begin at cursor location, scan backward (up)**
CHARS	—	**locate** *string-1* **as standalone or embedded in words**
PREFIX	—	**only look for** *string-1* **at the start of a word**
SUFFIX	—	**only look for** *string-1* **at the end of a word**
WORD	—	**only look for** *string-1* **as a standalone word**
X	—	**only search in lines that have been excluded**
NX	—	**only search in lines that have not been excluded**
col-1	—	**left column boundary to restrict search**
col-2	—	**right column boundary to restrict search**

CHANGE - Examples

```
CHANGE  PENGUINS  CAMELS

CHANGE  ALL  PENGUINS  HIPPOS

C  PENGUINS  DONKEYS  ALL   30   71

C  'Peace of mind'C    'Troubled in mind'C   FIRST

C  'trouble'C    'treble'C    NEXT

C  X'402021'   X'402120'    PREV

C  'END'    'ENDING'   SUFFIX  20  60

C  "LINCOLN'S DOCTOR'S DOG"  "LINCOLN'S DOCTOR'S FROG"  ALL

C   P'M<'   P'M>'   ALL
```

❒ **Both new and old strings must usually be delimited with the same character: blanks, quotes, or apostrophes**

❒ **You may use an unquoted asterisk for** *string-1* **or** *string-2* **(or both, if it makes sense)**

❒ **When picture strings are used:**

- **Both pictures must be of the same length**
- **Only the picture characters =, >, and < (meaning: leave unchanged, make uppercase, make lowercase, respectively) are allowed in the second string**

RFIND / RCHANGE - Edit / View Commands

RFIND	**—**	**Repeat last Find command**
RCHANGE	**—**	**Repeat last Change command**

- ♦ **RFIND usually assigned to PF5/PF17**
- ♦ **RCHANGE usually assigned to PF6/PF18**

Usage Technique:

❒ **If you want to change some but not all occurrences of a string,**

- ♦ **1. Position screen at desired starting location in file**
- ♦ **2. Enter the Change command on the command line, do not press <Enter>**
- ♦ **3. Press PF5; this will position the cursor to the first occurrence of the string to be changed**
- ♦ **4. To make the change, press PF6 then press PF5; to skip the change and go look at the next occurrence, press PF5**
- ♦ **5. Repeat step 4 until done**

RFIND / RCHANGE Wrap Around - and More

❒ **If FIND or CHANGE cannot find a match, the message "No CHARS *string-1* found" is issued**

❒ **If RFIND or RCHANGE are used and a match is not found, you'll see one of the messages "Top of data reached" or "Bottom of data reached", depending on the direction of the search**

❒ **If you issue the RFIND or RCHANGE command again, the search will "wrap around" to the other end of the data and continue on**

❒ **The direction of search is determined by the keywords in the previous FIND or CHANGE command:**

- ♦ **NEXT, ALL, and FIRST imply forward search (towards the end of the data)**
- ♦ **PREV and LAST imply backward search (towards the top of the data)**

❒ **After a CHANGE command has been issued, you may issue a FIND command and press F6 (RCHANGE)**

- ♦ **The result will be to change the next occurrence of the string on the FIND command to the value of the second string on the CHANGE command, for example:**

```
==> c mine yours     [press Enter]
==> f ours           [press F6]
```

- ♦ **Will change the next occurrence of** `ours` **to** `yours`
  - ✗ Although this behavior can be overridden using the EDSET command (discussed later) [z/OS 1.7 and later]

DELETE - Edit / View Primary Command

Syntax

```
DELETE      ALL      {X | NX}
```

ALL	**—**	**keyword**
X	**—**	**delete all lines that have been excluded**
NX	**—**	**delete all lines that have not been excluded**

❐ **DELETE ALL with no other operands is not accepted, as a precaution against error**

- ♦ **To delete all lines, whether they are excluded or not, you can code**

```
Command ===> del   all   nx;del   all   x
```

UNDO - Edit / View Primary Command

Syntax

```
UNDO
```

❒ **UNDO has no operands**

❒ **The last interaction with the editor is backed out**

- ♦ **"Interaction" means command entry or data change**

UNDO Requirements and Restrictions

- ♦ **Only changes made during the current editing of the data can be backed out and either UNDO or RECOVERY needs to be enabled**
- ♦ **The SAVE command clears out the undo buffer - you can only UNDO back to your most recent SAVE (or the starting point if you have not done any SAVEs)**
  - ✗ Except that in z/OS 1.9 or later, if UNDO is set as KEEP, the SAVE will save the data to disk but not clear out the undo buffer - so you can still issue an UNDO
- ♦ **UNDO affects only the data currently being edited**
- ♦ **The RECOVERY attribute must be ON or the UNDO attribute must be ON, KEEP, or RECOVER**
  - ✗ On the command line, enter: **RECOVERY ON**; ISPF will remember this whenever you edit [any member of] this data set
  - ✗ Or on the command line, enter: **SETUNDO** {**ON|KEEP|REC**}

❒ **If there are no more interactions to backout, a message is issued**

Computer Exercise: More Work With EDIT

In your training library, edit your member called PDFDATA1 and issue the comand RECOVERY ON (this enables UNDO, just in case); then:

1. Jot down any messages issued from the editor when you brought PDFDATA1 up to edit it:

   ____

   ____

2. Replace all leading lowercase letters to uppercase.

3. Replace all 'G' followed by two lowercase letters to 'S' (followed by the same two lowercase letters).

4. Replace all occurrences of lowercase 'or' to 'tor'.

5. Force all lines to be all uppercase (use the block line command UCC ... UCC), then issue the primary command: CAPS ON.

6. Replace all occurrences of 'FLELL' by 'FLY'.

7. Replace the first and every other occurrence of 'EE' by the string 'E ONLY' (that is, the 1st, 3rd, 5th, ... occurrences).

8. Replace all occurrences of 'KE' by 'THE'.

9. Replace all occurrences of 'LINGS' by ' THINGS'.

10. Replace all four-character long words where the last three characters are 'ELL' by the four-character long word 'TELL'.

11. How many occurrences are there now of the string 'NLY'? _____

12. Leave EDIT without saving PDFDATA1 as it now stands.

Section Preview

❒ **More on Edit / View**

- ♦ **Labels**
- ♦ **LOCATE — Edit / View Primary Command**
- ♦ **Edit / View Line Manipulation — Overlay**
- ♦ **Edit / View Line Manipulation — Text Split**
- ♦ **EDIT—Under—EDIT**
- ♦ **Edit / View / Browse and EPDF**
- ♦ **SORT — Edit / View Primary Command**
- ♦ **More on Edit / View (Machine Exercise)**

Labels

- ❐ **If you need to limit a search for a FIND, CHANGE, or EXCLUDE command to certain rows, you can do that by establishing LABELS in the data**

- ❐ **Labels simply tag particular records (lines) in the data while the data is in memory**

  - ♦ **Labels are not physically added to the data, nor are they stored with the data when the data is saved**

- ❐ **A label stays with a line even if the line is moved around in the editing process**

- ❐ **If the line is deleted, the label is deleted**

- ❐ **Labels can be globally removed by the RESET LABEL command:**

```
RESET  LABEL
```

Labels, 2

❐ **A label consists of a period followed by one to five alphabetic characters (no numerics or special characters), the first of which is not Z**

- ♦ **Note that ISPF uses labels beginning with Z, and has available the labels .ZFIRST and .ZLAST, which you cannot change but you may reference**

  - ✗ These represent the first and last lines in the data, respectively; these labels may also be specified as .ZF and .ZL

- ♦ **The ISPF label .ZCSR is also available; this represents the line on which the cursor is currently located**

❐ **Sample labels:**

.A

.FIRST

.SUBA

.SUBB

Labels, 3

❒ **Assign a label under Edit or View by typing the label over the sequence numbers on the display**

❒ **Then you can issue commands using labels:**

```
FIND    'NEW MEXICO'      .PARTA  .PARTB  FIRST
CHANGE      HISTORY     HERSTORY     .A    .BARK    ALL
EXCLUDE     P'>'       .OPEN        .CLOSE       ALL        15
DELETE      ALL        .ZFIRST      .JEND
```

❒ **Also, LOCATE in Edit / View can accept a label as an operand:**

```
LOCATE      .PLA
```

❒ **Possible application: place-holders for one or more lines while you examine different areas of the data**

LOCATE - Edit / View Primary Command

❐ **The LOCATE command, when not specifying a line number or label, may also indicate a particular line <u>type</u>:**

```
LOCATE     [FIRST]     {CHANGE}       [label range]
           [LAST]      {COMMAND}
           [NEXT]      {ERROR}
           [PREV]      {EXCLUDED}
                       {SPECIAL}
```

FIRST, LAST as before

NEXT, and PREV as before, except search begins at first line of current display, proceeding forwards or backwards, respectively

CHANGE	**—**	**line with CHANGE message (==CHG>)**
COMMAND	**—**	**line containing any line command**
ERROR	**—**	**line with error flag (==ERR>)**
EXCLUDED	**—**	**excluded line**
LABEL	**—**	**line containing any label**
SPECIAL	**—**	**line containing one of:**

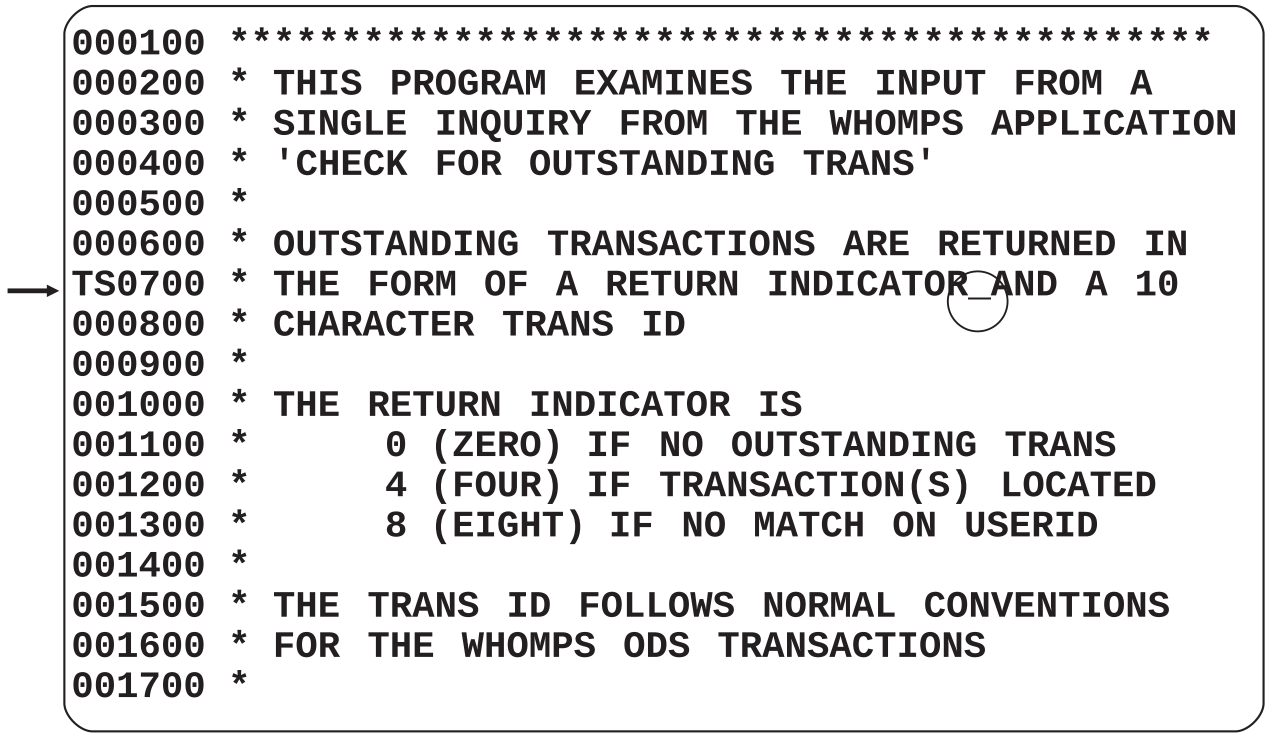
=BNDS>
=COLS>
======
=MASK>
==MSG>
=NOTE=
=PROF>
=TABS>

Edit / View Line Manipulation — Overlay

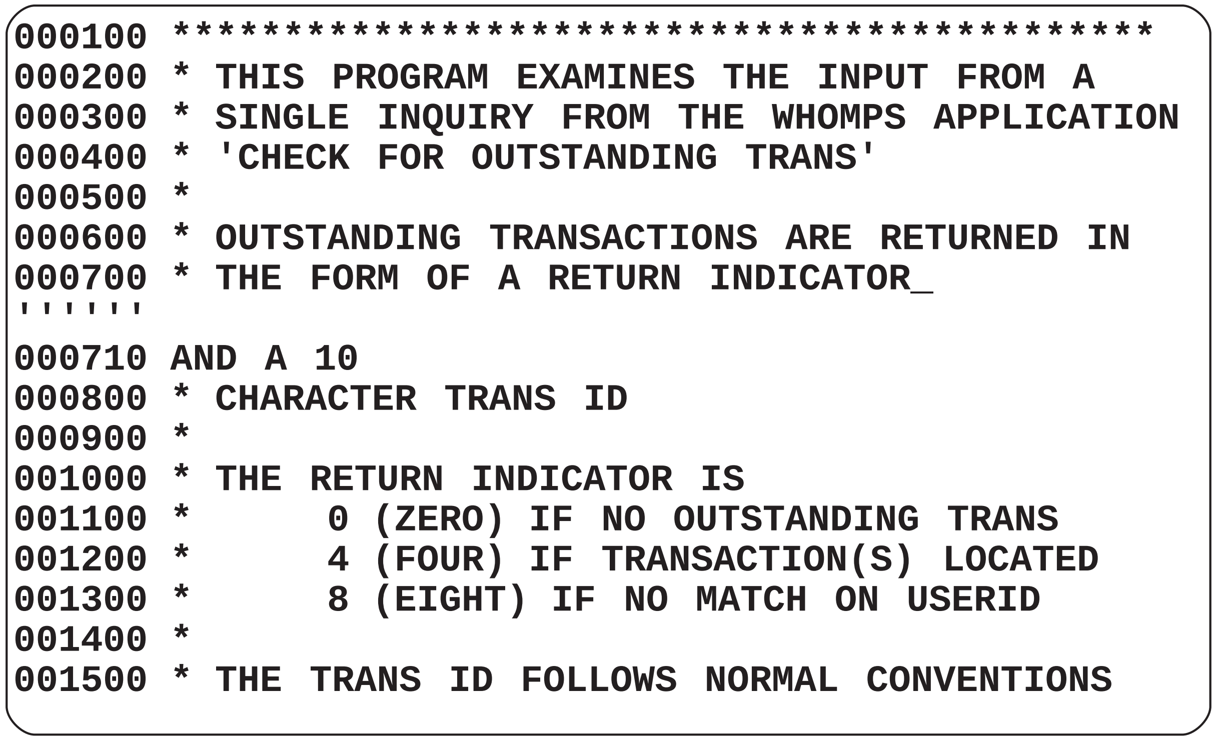
```
000100 PGM1     CSECT
OO0200          SAVE   (14,12)
000300          USING  PGM1,12
000400          LR     12,15
000500          ST     13,SAVE+4
000600          LR     2,3
000700          LA     13,SAVE
OO0800          ST     13,8(2)
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
MM0910                            SAVE REGISTERS
000920                            ESTABLISH
000930                                ADDRESSABILITY
000940                            STORE BACKWRD PTR.
000950                            SAVE ADDRESS OF SA
000960                            SET UP OWN SA
MM0970                            STORE FORWARD PTR
001000          BAL    3,GETDATE
```

```
000100 PGM1     CSECT
000200          SAVE   (14,12)    SAVE REGISTERS
000300          USING  PGM1,12    ESTABLISH
000400          LR     12,15          ADDRESSABILITY
000500          ST     13,SAVE+4  STORE BACKWRD PTR
000600          LR     2,3        SAVE ADDRESS OF SA
000700          LA     13,SAVE    SET UP OWN SA
000800          ST     13,8(2)    STORE FORWARD PTR
000900          OPEN  (MSTR,,TRANS,,EDTLST,(OUTPUT))
001000          BAL    3,GETDATE
001100          BAL    3,SETPARMS
001200          BAL    3,ATTCHSUB
001300          WTO    'INITIALIZATION COMPLETE'
001400          BAL    3,READTRAN
001500          OI     SW,1
001600 BIGLOOP  EQU    *
001700          TM     SW,1
```

Edit / View Line Manipulation — Text Split



```
000100 ********************************************
000200 * THIS PROGRAM EXAMINES THE INPUT FROM A
000300 * SINGLE INQUIRY FROM THE WHOMPS APPLICATION
000400 * 'CHECK FOR OUTSTANDING TRANS'
000500 *
000600 * OUTSTANDING TRANSACTIONS ARE RETURNED IN
TS0700 * THE FORM OF A RETURN INDICATOR_AND A 10
000800 * CHARACTER TRANS ID
000900 *
001000 * THE RETURN INDICATOR IS
001100 *      0 (ZERO) IF NO OUTSTANDING TRANS
001200 *      4 (FOUR) IF TRANSACTION(S) LOCATED
001300 *      8 (EIGHT) IF NO MATCH ON USERID
001400 *
001500 * THE TRANS ID FOLLOWS NORMAL CONVENTIONS
001600 * FOR THE WHOMPS ODS TRANSACTIONS
001700 *
```

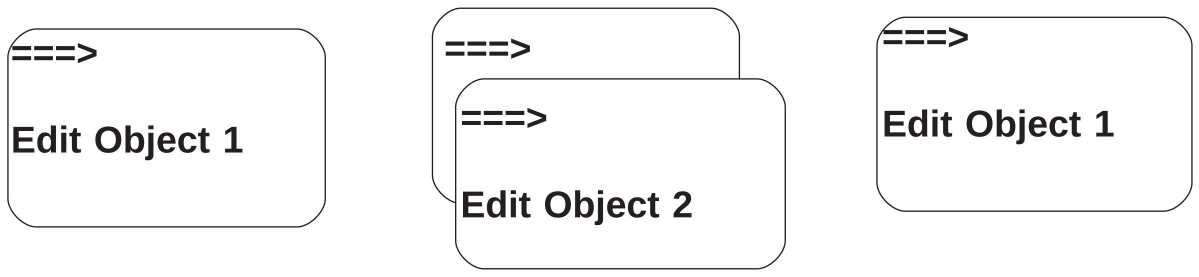


```
000100 ********************************************
000200 * THIS PROGRAM EXAMINES THE INPUT FROM A
000300 * SINGLE INQUIRY FROM THE WHOMPS APPLICATION
000400 * 'CHECK FOR OUTSTANDING TRANS'
000500 *
000600 * OUTSTANDING TRANSACTIONS ARE RETURNED IN
000700 * THE FORM OF A RETURN INDICATOR_
''''''
000710 AND A 10
000800 * CHARACTER TRANS ID
000900 *
001000 * THE RETURN INDICATOR IS
001100 *      0 (ZERO) IF NO OUTSTANDING TRANS
001200 *      4 (FOUR) IF TRANSACTION(S) LOCATED
001300 *      8 (EIGHT) IF NO MATCH ON USERID
001400 *
001500 * THE TRANS ID FOLLOWS NORMAL CONVENTIONS
```

EDIT - Under - EDIT

❒ **When you are in Edit, on the command line you can issue the EDIT primary command!**

❒ **This allows you to edit a different data set or member while holding your place in the first one**



❒ **When you press F3, or enter the END command, the second edit is terminated and you pick up where you left off in the previous edit**

EDIT - Under - EDIT, continued

❒ **If you wish to edit a member in the same library you are currently editing, you can type in:**

EDIT membername

❒ **If you wish to edit in a different library or sequential data set, just enter EDIT with no operand:**

EDIT

- ♦ **Then you will see the data set selection panel to specify the PDS or sequential data set you wish to edit**

❒ **There is no limit to the number of levels deep you can edit (although you can lose track of where you are)**

❒ **You cannot recursively edit the same member or data set, since EDIT requires exclusive allocation**

Edit / View / Browse

❒ To generalize the idea of Edit-under-Edit, the following commands are allowed:

- Under View you can issue: Edit, View, Browse commands
- Under Edit you can issue Edit, View, Browse commands
- If you are in Browse mode of View, you can issue Edit, View Browse commands

❒ The process is the same as discussed on the previous pages

Edit / View / Browse and EPDF

- ❐ **Alternatively, you can use the EPDF command to edit, view, or browse from <u>any</u> command or option line in ISPF**

  <u>Syntax</u>

  EPDF *dsname*[*(member_name)*] **[BROWSE | VIEW]**

  - ♦ **If you name a library and no *member_name*, you get the member list**

  - ♦ **If you omit BROWSE and VIEW, you get EDIT**

SORT - Edit / View Primary Command

❒ **You can sort the lines you are currently editing by coding the SORT command:**

```
SORT
```

♦ **Sorts all records using the entire record width as the sort field**

```
SORT   .AAX        .BBY
```

♦ **Sorts only those records between the (previously established) labels**

```
SORT   [ X | NX ]
```

♦ **Sorts all excluded records, or only non-excluded records**

```
SORT   A   10  15      D   3   7
```

♦ **Sorts all records on two sort fields, the first (primary) sort field is columns 10 - 15 inclusive, in ascending order; the second (minor) sort field is columns 3 - 7 inclusive, in descending order**

SORT - Edit / View Primary Command, 2

❒ **The full syntax of the sort command is**

```
SORT    [label range]  [X|NX]    [[ A | D ] [ col-1 ]  [ col-2 ] ... ]
```

- ♦ **You may specify up to five sort fields**
- ♦ **The sort fields may not overlap**

Computer Exercise: More on Edit / View

1. Get into View of your OLDPOEM member.
   a. Sort the data to be in ascending sequence by columns 2-5
   b. What happens when you try exit out (F3)?

2. Get into Edit of your OLDPOEM member.
   a. Sort the data to be in ascending sequence by columns 2-5
   b. What happens when you try exit out (F3)?

3. In your TRAIN.LIBRARY PDS, create a new member, PDFDATAZ. Key in the following line (note: the beginning /* should begin in column 2):

```
 /*  This exec emits rows to our Web server.      */
```

   Do a Save. Now, replace "emits" with "retrieves". Do a text split right before the word "to" and add "from the ADDRESS   */" to the first line. Insert these two lines after the first:

```
 /*  table, extracts specific values from these   */
 /*  rows and embeds these values into lines of   */
```

   Then finish up by inserting "/* XML that are then emitted " at the front of the remaining line. The result should look like this:

```
 /*  This exec retrieves rows from the ADDRESS    */
 /*  table, extracts specific values from these   */
 /*  rows and embeds these values into lines of   */
 /*  XML that are then emitted to our Web server. */
```

4. Before leaving this data, use the EPDF command to View member PDFDATA1.

Exit ISPF and logoff TSO.

Section Preview

❒ **Edit / View - Passing and Receiving Data**

- ♦ **CREATE, REPLACE, COPY, MOVE Edit Primary Commands**
- ♦ **CUT and PASTE Edit Primary Commands**
- ♦ **EDITSET (EDSET) Edit Primary Command**
- ♦ **Copy, Cut, and Paste in Edit (Machine Exercise)**

Edit / View - Passing and Receiving Data

❒ **Edit and view have some nice capabilities to allow you to combine, create, and otherwise modify members, data sets, and z/OS UNIX files**

❒ **The following primary commands work in similar styles:**

CREATE

take all or part of the data you are currently editing / viewing and place these lines in a library as a new member (if the member already exists, CREATE will fail) or create a sequential data set or z/OS UNIX file

REPLACE

take all or part of the data you are currently editing / viewing and place these lines into a library, replacing any previously existing member with that name (if no such member exists, it is created); the output of REPLACE may also be a sequential data set or a z/OS UNIX file

COPY

bring all or part of an existing member, sequential data set, or z/OS UNIX file into the data you are currently editing / viewing

MOVE

same as COPY, but delete the member, data set, or file that has been moved

CREATE - Edit / View Primary Command

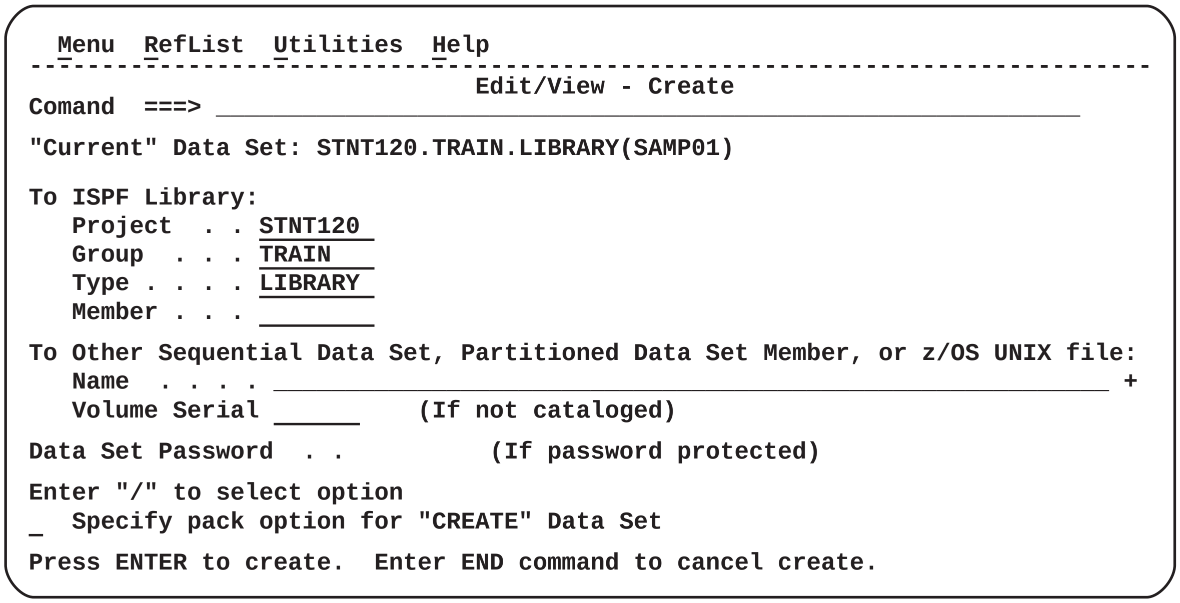
❒ **Indicate which line(s) you want to copy or move into the new member, data set, or file; use the C or M line commands:**

```
→ C99900 IN THIS EXAMPLE, ALL LINES WILL BE COPIED INTO
  000200 THE NEW MEMBER (IF 999 OR FEWER LINES HERE)
```

❒ **Then issue:**

Command ===> create

- **This will display a panel to enter the target library and member name, sequential data set name, or z/OS UNIX file name, like this:**



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Edit/View - Create
Comand  ===> ____________________________________________________________

"Current" Data Set: STNT120.TRAIN.LIBRARY(SAMP01)

To ISPF Library:
   Project  . . STNT120
   Group  . . . TRAIN
   Type . . . . LIBRARY
   Member . . . ________

To Other Sequential Data Set, Partitioned Data Set Member, or z/OS UNIX file:
   Name  . . . . __________________________________________________________ +
   Volume Serial ______    (If not cataloged)

Data Set Password  . .         (If password protected)

Enter "/" to select option
_  Specify pack option for "CREATE" Data Set

Press ENTER to create.  Enter END command to cancel create.
```

- **The identified lines will be placed in the target member**

- **When the target is a member in a PDS, if a member by this name exists, the CREATE will fail (use REPLACE)**

CREATE - Edit / View Primary Command, 2

```
Command ===> create .first .last
```

- **Works the same way, but the source data is identified by labels you've previously put in the data**

```
Command ===> create membername
```

- **Will create a new member in the same library you are currently editing / viewing; you won't see a panel to enter the library name**

```
Command ===> create membername .first .last
```

- **Works as you would expect from the above discussion**

❒ **So "Create" copies or moves lines from data being edited / viewed into a member of a library (PDS or PDSE), a sequential data set or a z/OS UNIX file**

- **Which lines, and whether you are copying or moving, are identified by line commands**
- **If the target member is in the same library you are currently working in, code CREATE membername**
- **Otherwise, code CREATE and then fill in the blanks in the resulting screen**

CREATE - Edit / View Primary Command, 3

❒ **You can specify the target of a CREATE command as:**

♦ *membername*	**(as discussed before)**
♦ **(***membername***)**	**(the closing parenthesis is optional)**
♦ *dsn***(***membername***)**	
♦ *dsn*	**(a sequential data set name)**

- ✗ Where *dsn* is either fully-qualified and quoted or it will be prefixed by your userid
- ✗ If *dsn* doesn't exist, you will be prompted to have it allocated for you

♦ *z/OS_UNIX_file_name*	**(a z/OS UNIX path+file)**

REPLACE - Edit / View Primary Command

❒ **REPLACE works the same as CREATE except:**

- ♦ **If the object is a sequential data set, it must already have been allocated or you will be prompted to allocate it right then**
- ♦ **If the target object already exists, it will be replaced by the specified line(s)**
- ♦ **However, if a member is specified and it does not exist, the member will be created in the specified PDS**

❒ **You can specify the target of a REPLACE command to be:**

- ♦ *membername*
- ♦ **(***membername***)**
- ♦ *dsn*
- ♦ *dsn***(***membername***)**
  - ✗ Where *dsn* is either fully-qualified and quoted or it will be prefixed by your userid
  - ✗ If *dsn* does not exist, you will be prompted to have it allocated for you
- ♦ *z/OS_UNIX_file_name*

COPY - Edit / View Primary Command

❒ **If the data set, member, or z/OS UNIX file you are editing or viewing is not empty, you need to indicate where you want the copied data to go: use the line command A or B:**

```
  000100 THIS IS JUST A TEST DATA LINE
→ A00200 AS IS THIS LINE
  000300 THE COPIED DATA WILL BE PUT ABOVE THIS LINE
```

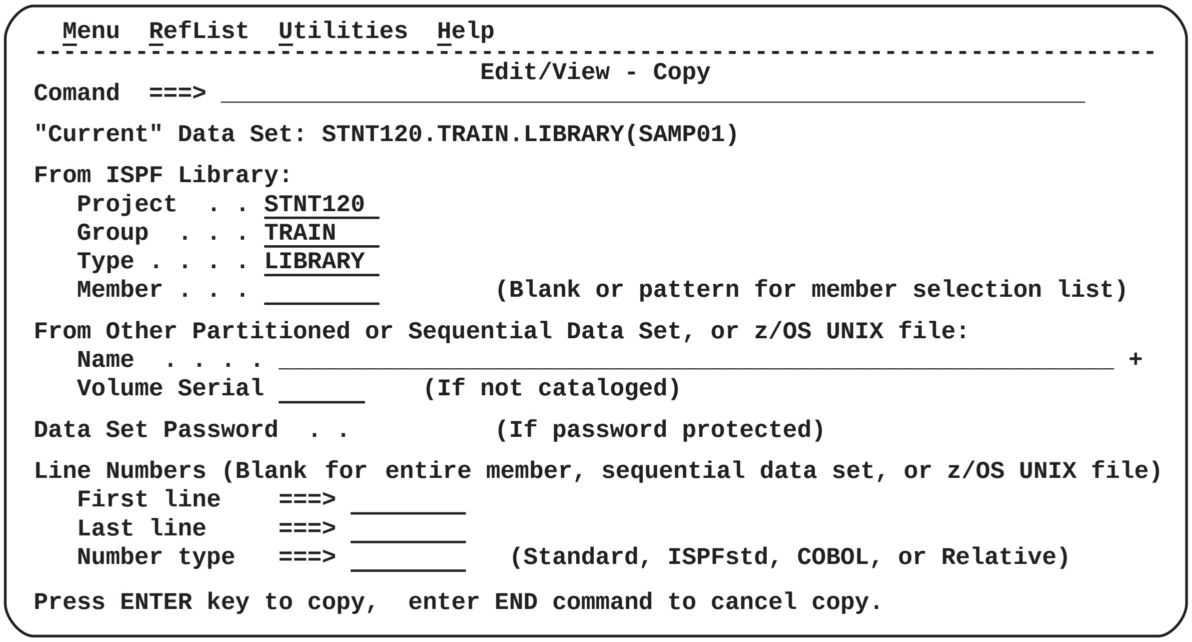
❒ **Then, on the command line, issue:**

Command ===> copy

✗ This will get you a panel to enter the target name

✗ If you specify a library, you may also specify a member name, or leave that part blank and get a member selection list

✗ Notice you may copy all lines or just a range of lines:



```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                              Edit/View - Copy
Comand  ===> ____________________________________________________________

"Current" Data Set: STNT120.TRAIN.LIBRARY(SAMP01)

From ISPF Library:
   Project  . . STNT120
   Group  . . . TRAIN
   Type . . . . LIBRARY
   Member . . . ________        (Blank or pattern for member selection list)

From Other Partitioned or Sequential Data Set, or z/OS UNIX file:
   Name  . . . . __________________________________________________________ +
   Volume Serial ______     (If not cataloged)

Data Set Password  . .          (If password protected)

Line Numbers (Blank for entire member, sequential data set, or z/OS UNIX file)
   First line    ===> ________
   Last line     ===> ________
   Number type   ===> ________   (Standard, ISPFstd, COBOL, or Relative)

Press ENTER key to copy,  enter END command to cancel copy.
```

♦ **The identified data will be placed where you indicated**

COPY - Edit / View Primary Command, 2

Copy *membername*

- **Will copy a member if it is in the same library you are currently working in; you won't see a panel to enter the library name**

Copy [after | before] .label

and

Copy *membername* [after | before] .label

- **Work as you might expect, specifying where to put the incoming data relative to a label you've already established**

❒ **So "Copy" brings in lines to the data being edited / viewed from a member of a library (PDS or PDSE), a sequential data set, or a z/OS UNIX file**

- **Where to put the incoming lines is indicated by the A or B line commands**

  ✗ If you are editing an empty screen, you do not need A nor B

- **Which lines are identified by line numbers in the subsequent panel**

- **If the source member is in the same library you are currently editing in, code COPY membername**

- **Otherwise, code COPY and then fill in the blanks in the resulting panel**

COPY - Edit / View Primary Command, 3

❒ **You can specify the source data for a COPY command as:**

- ♦ *membername*
- ♦ **(***membername***)**
- ♦ *dsn*
- ♦ *dsn***(***membername***)**
  - ✗ Where *dsn* is either fully-qualified and quoted or it will be prefixed by your userid
- ♦ *z/OS_UNIX_file_name*

MOVE - Edit / View Primary Command

❒ **MOVE works the same as COPY, except the source data is deleted after the copy part is done**

```
          Command ===> move
      and
          Command ===> move source_name
      and
          Command ===> move   [after | before]   .label
      and
          Command ===> move source_name   [after | before]   .label
```

- ♦ **All work as you would expect**
  - ✗ For non-empty targets, you need to indicate where to put the data, using A or B or labels

❒ **You can specify the source data for a MOVE command as:**

- ♦ *membername*
- ♦ **(***membername***)**
- ♦ *dsn*
- ♦ *dsn***(***membername***)**
  - ✗ Where *dsn* is either fully-qualified and quoted or it will be prefixed by your userid
- ♦ *z/OS_UNIX_file*

CREATE, REPLACE, MOVE, and UNDO

❐ **UNDO affects only the data currently being edited or viewed, not an external member, data set, or z/OS UNIX file**

Hence

- ♦ **UNDO cannot backout a CREATE or REPLACE**
  - ✗ Except that if you had used M line commands to put data out, those lines will be restored
- ♦ **UNDO cannot restore the original data for a MOVE comand**
  - ✗ Except that lines brought in to the current data can be deleted (and this works for both COPY and MOVE)

Cut and Paste Concepts

❐ **There are many times when it would be convenient to copy a range of lines from a file you are editing or viewing and paste them into another file**

- ♦ **So far, the choices you have are these:**

  - ✗ Use CREATE to put the lines into a file; get into edit of the target; use MOVE or COPY to retrieve the lines

  - ✗ Use your terminal emulator's capability of using the mouse and emulator commands for the copy and paste

Cut and Paste Concepts, 2

❒ **The ISPF editor has CUT and PASTE commands that let you do similar operations**

- ♦ **For CUT, indicate a range of lines using line commands:**
  - ✗ C, M, CC...CC, or MM...MM
- ♦ **or labels**
  - ✗ User labels or ISPF-supplied labels: .zfirst, .zlast, .zcrsr
- ♦ **place the cut lines into the default, unnamed clipboard, or into a named clipboard (you create using the name)**
  - ✗ REPLACE-ing data that was already there or APPEND-ing to data that was already there
- ♦ **The area for holding cut lines, the clipboard, is implemented using data spaces, giving more power than any previous alternative (in terms of capacity and number of clipboards)**
  - ✗ Maximum number of clipboards is 10 (in addition to DEFAULT), although installation standards can reduce the number and size of the clipboards
- ♦ **You can cut all lines, all excluded lines, or all non-excluded lines**

Cut and Paste Concepts, 3

- **For PASTE, select any available clipboard and place the lines into the target**
  - **After or before a line (using "a" or "b" line commands)**
  - **After or before a label (using user labels or ISPF labels)**
  - **KEEPing or DELETE-ing the lines from the clipboard**

CUT Command

CUT - syntax

CUT **[***line_range***] [***clipboard_name***] [REPLACE | APPEND] [X|NX]**

- **place the lines indicated into the clipboard**

  - ✗ line_range may be a starting and ending line number:
    ```
    cut   22  130
    ```
  - ✗ line_range may be labels:
    ```
    cut   .zfirst  .zcrsr
    ```
  - ✗ line_range may be a starting line number and ending label or a starting label and an ending number:
    ```
    cut   22   .zlast
    cut   .zcrsr   324
    ```
  - ✗ If line_range is omitted, and you do not have line commands **c**, **m**, **cc...cc** or **mm...mm** already placed, <u>the entire file is cut</u>

- **if no name is specified, the DEFAULT clipboard is used**

- **If a name is specified and no clipboard by that name exists, one is created**

- **If REPLACE is specifed, any existing lines in the clipboard are replaced by the cut lines**

- **If APPEND is specified, the cut lines are appended to the existing lines in the clipboard**

- **If X is specified, only cut excluded lines; NX cuts only non-excluded lines; if neither is coded, all lines are cut**

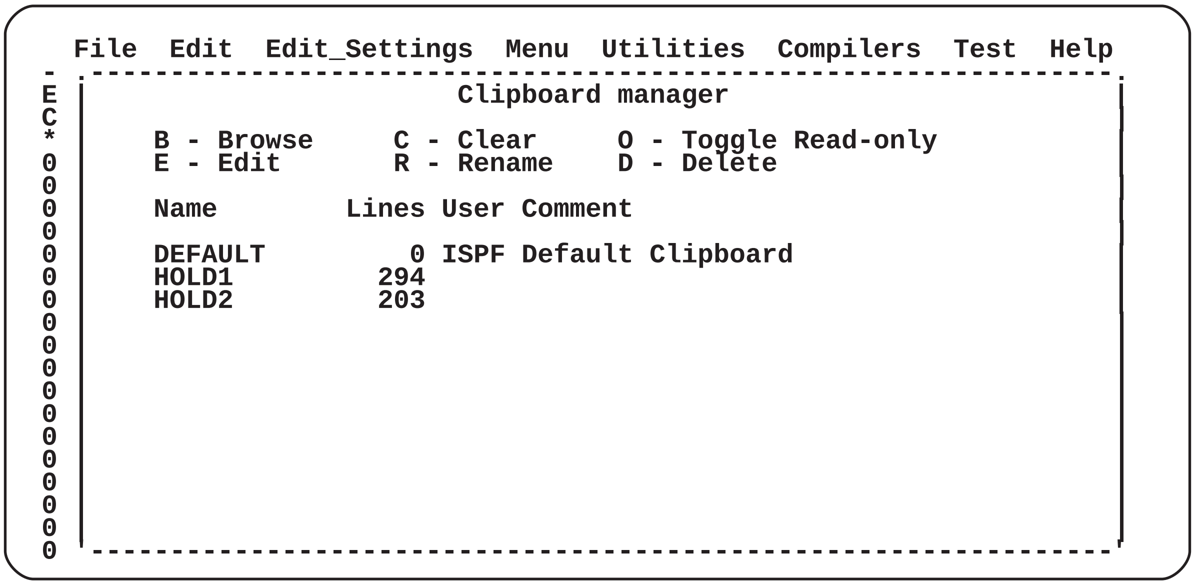
CUT Command, 2

CUT - syntax, continued

CUT DISPLAY

- ♦ **Display the current list of clipboards and how many lines they contain**

- ♦ **You may also examine the clipboard contents:**



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
- .----------------------------------------------------------------.
E |                      Clipboard manager                          |
C |                                                                 |
* |     B - Browse     C - Clear     O - Toggle Read-only           |
0 |     E - Edit       R - Rename    D - Delete                     |
0 |                                                                 |
0 |     Name        Lines User Comment                              |
0 |                                                                 |
0 |     DEFAULT         0 ISPF Default Clipboard                    |
0 |     HOLD1         294                                           |
0 |     HOLD2         203                                           |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 |                                                                 |
0 '----------------------------------------------------------------'
```

PASTE Command

<u>PASTE - syntax</u>

PASTE **[***clipboard_name***] [{AFTER | BEFORE}** *label***]**

[KEEP | DELETE]

- **If clipboard_name is omitted, the default clipboard is assumed**
- **If AFTER or BEFORE is specified, label can be a user label or an ISPF label**
- **If AFTER or BEFORE is not specified, you must have an "a" or a "b" in the line number field to indicate where the lines are to go**
- **If KEEP is specified, the copied lines remain in the clipboard**
- **If DELETE is specified, the copied lines are deleted from the clipboard**

The EDITSET Command

❒ **The EDITSET primary command lets you set various edit profile defaults**

- **Although this was primarily designed for changing the default behaviors of CUT and PASTE it has some wider applicabilty**

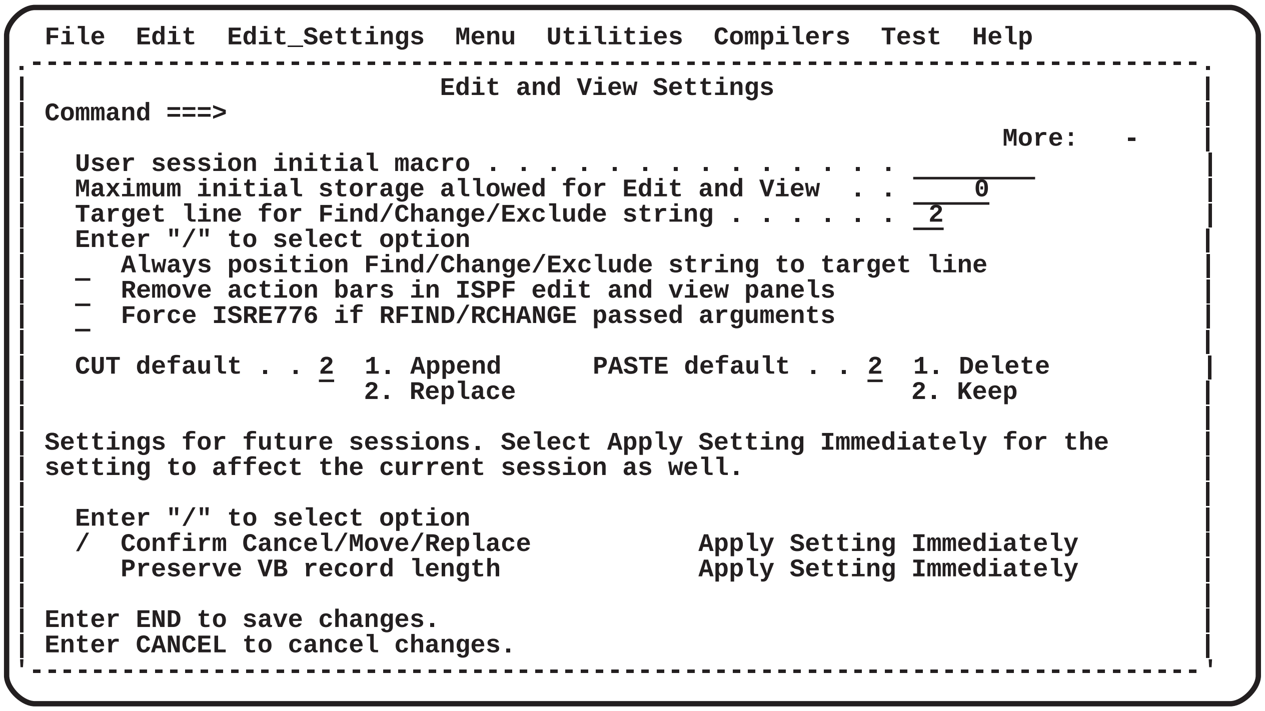
Syntax

EDITSET

or the synonym

EDSET

- **There are no operands, and the command displays a panel like this:**



```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
.----------------------------------------------------------------------.
|                          Edit and View Settings                      |
| Command ===>                                                         |
|                                                           More:   -   |
|   User session initial macro . . . . . . . . . . . . . . _________   |
|   Maximum initial storage allowed for Edit and View  . .      0      |
|   Target line for Find/Change/Exclude string . . . . . .  2          |
|   Enter "/" to select option                                         |
|   _  Always position Find/Change/Exclude string to target line       |
|   _  Remove action bars in ISPF edit and view panels                 |
|   _  Force ISRE776 if RFIND/RCHANGE passed arguments                 |
|                                                                      |
|   CUT default . . 2  1. Append      PASTE default . . 2  1. Delete   |
|                      2. Replace                           2. Keep     |
|                                                                      |
| Settings for future sessions. Select Apply Setting Immediately for the |
| setting to affect the current session as well.                       |
|                                                                      |
|   Enter "/" to select option                                         |
|   /  Confirm Cancel/Move/Replace           Apply Setting Immediately |
|      Preserve VB record length             Apply Setting Immediately |
|                                                                      |
| Enter END to save changes.                                           |
| Enter CANCEL to cancel changes.                                      |
'----------------------------------------------------------------------'
```

The EDITSET Command, 2

❒ **Options on the panel ...**

- User session initial macro **- specify an edit macro to run in addition to any IMACRO (initial macro) you may have in your profile**
- Maximum initial storage **- value of 0 means no limits; any other integer represents KB (kilobytes) and a value is rounded up to 128KB multiple**
- Target line for found/changed/excluded string **- if you issue one of these commands, what line on the screen should it be positioned on (1-99; if value greater than number of lines on display, line positioned at last line on display)**
- Always position found/change/excluded string to target line **- if selected, a found line will always be positioned as specified above; if not selected, found lines are not repositioned if they are already visible on the display**
- Force ISRE776 if RFIND/RCHANGE passed arguments **- if you issue either RFIND or RCHANGE (or press a function key assigned to those), ISPF should issue a message if data on command line (and ignore the command line contents)**
- Remove action bars in ISPF edit and view panels **- gives you some extra screen "real estate" while editing**
  - ✗ Note this will hold for all edit and view sessions using ISPF-supplied panels, and will be remembered across logons!
- **Also, you can set the** CUT default **and** PASTE default **actions**
- **Also, you can** turn on / off confirmation panels **when issuing Cancel, Move, or Replace commands, and tell the editor to** preserve trailing spaces **(or not) for VB records**

Computer Exercise: Copy, Cut, and Paste in Edit

1. In your <userid>. TRAIN.LIBRARY data set, edit a new member, called TRF2F.

   a. Copy in member BASE from your library

   b. Copy in member INITIALZ from your library, right before the last line already in the data you're editing

   c. Copy in member LOGIC from your library, right before the last line already in the data you're editing

   d. Copy in member TERMINAT from your library, right before the last line already in the data you're editing

   e. Copy in member DATAREAS from your library, right before the last line already in the data you're editing

   f. Copy in member FILES from your library, right before the last line already in the data you're editing

   You now have constructed a small, but complete, program, written in Assembler language. Look at it briefly to get a "feel" for it. [You are not expected to become an Assembly language programmer at this point.]

2. In your new member, TRF2F, there are three comment blocks (lines 10-14, 19-23, and 29-33). Using Cut, Paste, and any other necessary ISPF edit skills, work so that you end up with these members:

   TRF2F - as just constructed in 1) above

   TRF2FNC - TRF2F without any comment blocks

   TRCOMS - all the comment lines from TRF2F.

   During your work, use CUT DISPLAY to examine your clipboard(s).

3. Use the EDSET command to establish your preferred defaults for CUT, PASTE and other edit options.

This page intentionally left almost blank.

Section Preview

❒ **Edit Profiles**

- ♦ **Profile options**
- ♦ **Building, saving, using profiles**
- ♦ **Bounds, Mask, Tabs lines**
- ♦ **Edit Action Bar Choices**
- ♦ **Language-sensitive Color Editing**
- ♦ **MD, VERSION, LEVEL commands**
- ♦ **Recent Edit / View Line commands: AK, BK, OK, HX**
- ♦ **Using Tabs and other Profile Characteristics (Machine Exercise)**

Edit Profiles

❒ **When you are editing a sequential data set or member of a PDS, you are working under an edit profile**

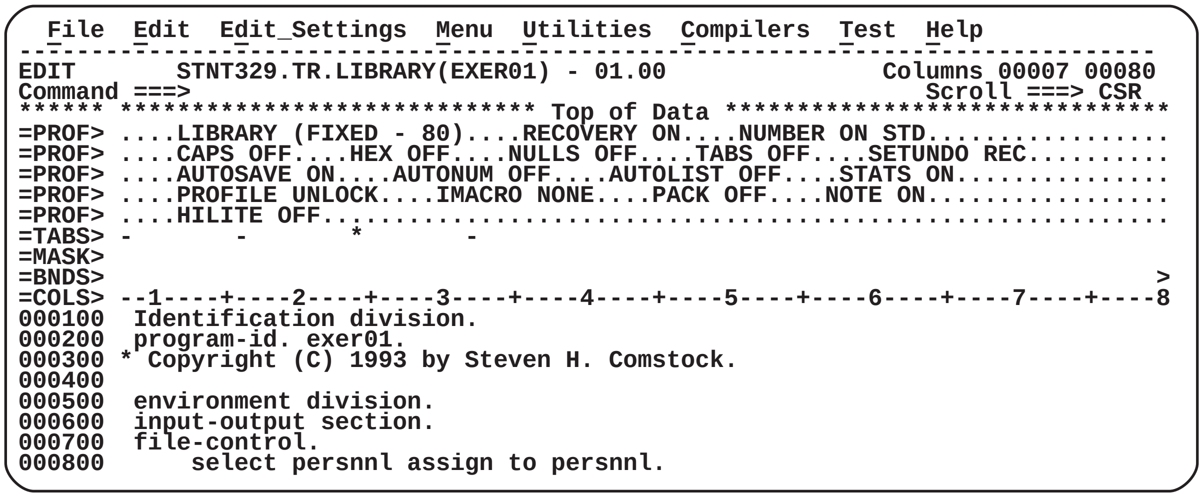
❒ **An edit profile is a collection of characteristics and attributes that describes how you want to work with this data**

- **For example, you may have the edit profile attribute of "CAPS ON" established**

  ✗ In this case, all alphabetic characters will automatically be translated to uppercase when you type in the data

  ➢ Regardless of how your terminal keyboard is set and how you use the shift key

❒ **You can display the current profile settings by issuing the Profile command from the Command / Option line; the settings lines are inserted in the display like this:**



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       STNT329.TR.LIBRARY(EXER01) - 01.00               Columns 00007 00080
Command ===>                                                   Scroll ===> CSR
****** ***************************** Top of Data ******************************
=PROF> ....LIBRARY (FIXED - 80)....RECOVERY ON....NUMBER ON STD................
=PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF....SETUNDO REC.........
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON..............
=PROF> ....PROFILE UNLOCK....IMACRO NONE....PACK OFF....NOTE ON................
=PROF> ....HILITE OFF..........................................................
=TABS> -       -       *       -
=MASK>
=BNDS>                                                                        >
=COLS> --1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
000100  Identification division.
000200  program-id. exer01.
000300 * Copyright (C) 1993 by Steven H. Comstock.
000400
000500  environment division.
000600  input-output section.
000700  file-control.
000800      select persnnl assign to persnnl.
```

❒ **The major edit profile attributes are discussed on the following pages**

Edit Profile Options

AUTOSAVE	**ON —**	**automatically save changes made when exit from Edit**
	OFF —	**do not automatically save changes made when exit from Edit**
		PROMPT - at exit from Edit ask if want to save changes
		NOPROMPT - at exit from Edit automatically cancel changes
AUTONUM	**ON —**	**automatically renumber data when it is saved (NUMBER must be ON)**
	OFF —	**do not automatically renumber data when it is saved**
AUTOLIST	**ON —**	**if data saved and has been changed, automatically print a copy of the new version (send to LIST file)**
	OFF —	**do not do this**
CAPS	**ON —**	**translate alpha characters to upper case**
	OFF —	**accept all data as entered**
HILITE	many**—**	**for language sensitive editing; discussed later**

Edit Profile Options, 2

HEX	**ON —**	**display data in hexadecimal format**	
		VERT	**- two digits in vertical form**
		DATA	**- two digits in horizontal form**
	OFF —	**display data in character format**	
NULLS	**ON —**	**replace trailing blanks with NULLs until data saved**	
		STD	**non-empty field with trailing space(s) on screen as one blank followed by nulls; all-blank (empty) fields on screen as blanks**
		ALL	**all trailing blanks on screen as all nulls; empty fields on screen as all nulls**
	OFF —	**leave trailing blanks as blanks**	
NUMBER	**ON —**	**maintain sequence numbers in data**	
		STD	**last 8 columns for fixed length records, first 8 for variable length records**
		COBOL	**first 6 columns (fixed length records only)**
	(both of these can be in effect for fixed record lengths)		
	OFF —	**do not maintain sequence numbers in this data**	
PACK	**ON —**	**compress data when you save it**	
	OFF —	**save data as is**	

Edit Profile Options, 3

RECOVERY	ON —		keep backup of changes on disk until data is SAVEd or CANCELed
	OFF —		do not backup changes (performance improvement)
		WARN	— these options remain for — compatibility with earlier
		NOWARN	— releases only
SETUNDO	STORAGE —		save changes for UNDO processing in storage (may code as ON)
	KEEP —		do not flush undo buffers when SAVE issued
	RECOVERY—		save changes for UNDO processing on DASD; sets RECOVERY ON
	OFF —		allow saving of changes on DASD (if RECOVERY is ON) but not in storage
STATS	ON —		keep creation / update / size statistics
	OFF —		do not keep these statistics
	EXT —		keep extended statistics
TABS	ON —		activate tab character recognition; [specify logical tab character you wish to use]
		STD —	activate hardware tabs by inserting attribute characters in data at hardware tab positions where current data is blank or null
		ALL —	activate hardware tabs by inserting attribute characters in data at hardware tab positions regardless of data in those positions
	OFF —		turn tabs mode off; deactivates logical tabs and hardware tabs

Special Profile Lines

BOUNDS

- ♦ limits scope of these primary commands: CHANGE, EXCLUDE, FIND, SORT and these line commands: <, >, (,), TS

MASK

- ♦ pre-fill new lines with pre-determined character string

TABS

- ♦ contains tab settings

Establishing Edit Profiles

❒ **The editor selects default profile attributes based on the last level of the data set name**

- ♦ **Unless, on entry to edit you fill in the field**

  ```
  PROFILE NAME  ===>
  ```

- ♦ **If there is a profile of the specified or implied name, it is used; if there is not, a new profile is created; the initial values depend on several factors**

  - ✗ If there is a profile named ZDEFAULT, its values are used (you can create your own ZDEFAULT)

  - ✗ If there is no ZDEFAULT profile...

    - ➢ For ISPF 4.2 and earlier, there is a hard-coded set of defaults

    - ➢ For ISPF 4.2.1 and later, an installation-specified site-wide set of values is used

❒ **Once you are editing some data, you may change characteristics of your current profile by issuing primary commands:**

```
NUM ON STD
RECOVERY ON
NULLS ON
```

- ♦ **And so on**

Locking Edit Profiles

- ❐ **Once you have the mix of characteristics you want, you can issue the primary command**

```
PROFILE LOCK
```

  - ♦ **Your current profile is saved as is**
  - ♦ **You can change attributes of the profile while you are editing**
    - ✗ But this only resets the copy of your profile in memory
  - ♦ **To reset characteristics of your <u>saved</u> profile, if your profile is locked, you must first issue**

```
PROFILE UNLOCK
```

Saving Edit Profiles

❒ **You can name and save an edit profile for later use; just issue**

```
PROFILE   name
```

- ♦ **If *name* already exists, the profile by that name is brought in and established as your current edit profile**

- ♦ **If *name* does not already exist, the current set of characteristics is saved as a profile called *name***

❒ **If you issue the PROFILE command with no operands, the editor will display your current profile attributes at the top of your panel**

```
PROFILE
```

- ♦ **The lines of profile attributes are just for display; they are not put into your data (RESET removes the lines from the display)**

- ♦ **Also, coding PROFILE with an integer value displays additional lines (TABS, MASK, BOUNDS, and COLS lines), making a total of up to 9 lines:**

```
PROFILE   9
```

Rules for Profile Names

❒ 1 to 8 alphanumeric characters, first of which is alpha

❒ Do not use "LOCK" nor "UNLOCK" as profile names

❒ The number of profiles available to you is an installation chosen number between 1 and 255; the default is 25 distinct profile names per userid

- ♦ There is no way to get a list of available profile names
- ♦ There is no way to delete a particular profile

❒ Different profiles may be appropriate for different types of data, for example source code versus JCL versus data

Profile Notes

❒ **The editor may dynamically change these profile attributes**

  <u>CAPS</u>

  <u>NUMBER</u>

  <u>PACK</u>

  <u>STATS</u>

- ♦ **Depending on the data, at the time you start to edit**
- ♦ **If it does this, a warning message is issued**
- ♦ **At this point, you may change the profile characteristics back by issuing the appropriate primary command(s)**

Profile Notes, 2

❒ **For example, suppose you are editing some data, using the profile attribute CAPS OFF**

❒ **If you leave this data and go to edit some data that only has uppercase characters, the editor will say**

```
CHARACTERISTICS CHANGED TO CAPS ON FROM CAPS OFF;
DATA CONTAINS ONLY UPPERCASE CHARACTERS
```

- ♦ **Or words to this effect**
  - ✗ This is because the editor always scans each member or dataset as it is brought into memory for editing, checking for sequence numbers and upper/lowercase characters
  - ✗ Information about data packing and statistics is also examined and compared with the current profile settings

❒ **The RESET primary command deletes the message from the display**

- ♦ **Now, if you want the data you're editing to accept lowercase letters without translating to uppercase, you must issue**

```
CAPS OFF
```

Numbering Primary Commands

NONUMBER

- ♦ **Turns off sequence numbering (will not remove existing numbers if any present)**

NUMBER [OFF |ON {STD | COBOL | STD COBOL}]

- ♦ **Establishes number mode; inserts appropriate sequence numbers if not present (will not remove existing numbers if specified as NUMBER OFF)**

UNNUMBER

- ♦ **Removes sequence numbers from data**

RENUM [OFF |ON {STD | COBOL | STD COBOL}]

- ♦ **Renumbers sequence numbers in data; that is, starts with 100 and increments by 100 in either the Standard or the COBOL location (or both)**

Bounds Line

❒ **You can restrict the application of certain commands by establishing <u>bounds</u> (beginning and ending column numbers) as part of your current profile**

- ♦ **Either use the BOUNDS primary command:**

  BOUNDS [*left-col***] [***right-col***]**

- ♦ **Or the BOUNDS line command (type BOUNDS over sequence numbers), which displays the contents of the current bounds line**

  - ✗ You can then indicate the bounds you want by typing in < and > for the left and right boundaries, respectively

  - ✗ For example:

```
=COLS>----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
=BNDS>                    <                             >
```

❒ **Now all commands that can be restricted to column ranges (such as FIND, CHANGE, EXCLUDE, and so on) will be implicitly restricted to the columns in this range (explicit column ranges will override the boundaries if necessary)**

Mask Line

❒ You can establish an initial value for every new line by placing that value in the mask line

✗ For example, if you are coding a PL/I source program and you want a comment area to be included for every line, fill in the mask line as follows:

```
=COLS>----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
=MASK>                                      /*                     */
```

❒ Now, whenever you insert new lines (using the "I" line command) these new lines will not contain blanks but the mask that you defined

Tabs

❒ **Software tabs**

- **Represented in the TABS line by _ and - (underscore and hyphen)**
  - ✗ When <Enter> key is pressed, cursor is positioned to the tab location
  - ✗ Software tabs always activated if defined in the TABS profile line

❒ **Hardware tabs**

- **Represented in the TABS line by * (asterisk)**
  - ✗ Tab —> key positions cursor to position after the next hardware tab
  - ✗ Tab <— key positions cursor to position after previous hardware tab
  - ✗ Hardware tabs are activated if defined in the TABS profile line and if TABS ON is established in the edit profile characteristics
    - ➢ TABS ON is the same as TABS ON STD: the editor inserts attribute bytes that cannot be typed over in all tab positions containing blanks or nulls; tab positions already containing non-blank, non-null data will not be activated
  - ✗ Issuing TABS ON ALL causes attribute bytes to overlay all tab positions regardless of their current content, hiding existing data (TABS OFF will re-show the data)

Tabs, 2

❒ **Logical tabs**

- **Use hardware tab locations for positioning and identify a tab character using the primary command**

  TABS ON %

- **Choose your own logical tab character (% in the example above):**

  ✗ Not a number, letter, or command delimiter character; and do not use a character that may be found in the data you are going to enter

- **Type in data, throwing in logical tab characters where appropriate**

- **When you press <Enter>, the data will be tabbed to the hardware tab locations; for example**

```
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
=TABS>      *        *                   *
```

- **If you type in:** **%%SAMPLE%DATA**

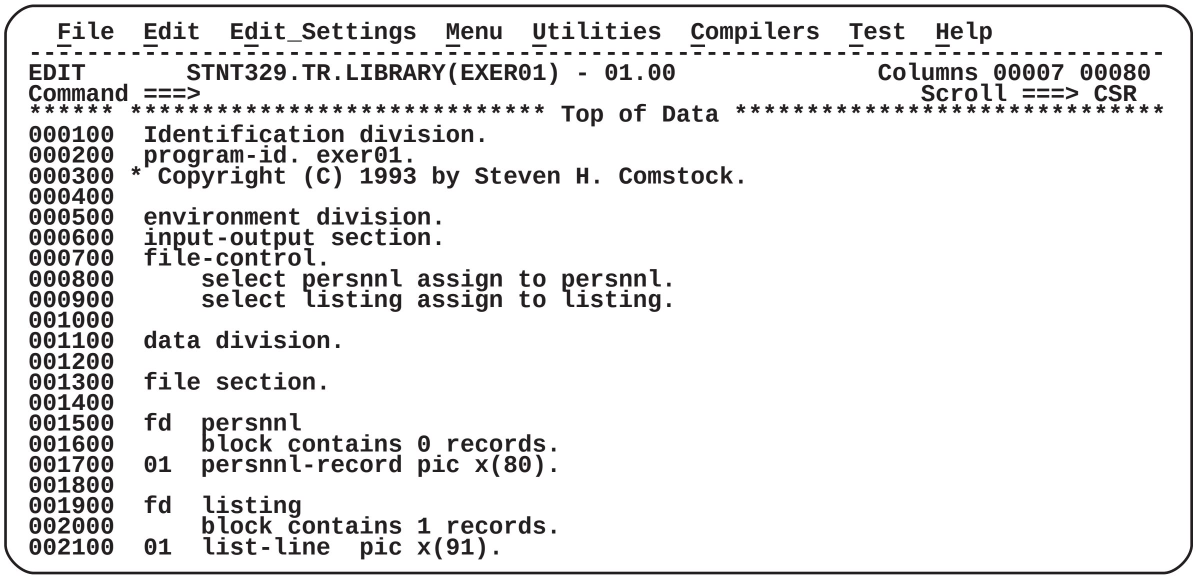
  ✗ Then when you press <Enter>, "SAMPLE" will begin in column 16 and "DATA" in column 36

- **Note that hardware tabs don't work if logical tabs are active**

Edit Panel Action Bar Choices

❒ We need to discuss the choices available from the Edit (and View) Action Bar

- ♦ Recall, the Edit panel looks like this:



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       STNT329.TR.LIBRARY(EXER01) - 01.00              Columns 00007 00080
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100  Identification division.
000200  program-id. exer01.
000300 * Copyright (C) 1993 by Steven H. Comstock.
000400
000500  environment division.
000600  input-output section.
000700  file-control.
000800      select persnnl assign to persnnl.
000900      select listing assign to listing.
001000
001100  data division.
001200
001300  file section.
001400
001500  fd  persnnl
001600      block contains 0 records.
001700  01  persnnl-record pic x(80).
001800
001900  fd  listing
002000      block contains 1 records.
002100  01  list-line  pic x(91).
```

❒ The Menu, Utilities, and Help choices are the same as we have already seen

❒ The Compilers and Test options are beyond the scope of this course

❒ The Edit_Settings choice gives you only one choice: Edit Settings, which basically invokes the EDSET command (discussed earlier)

Edit Panel Action Bar Choices, continued

❒ The File pull-down has three choices

- ♦ Save - execute the Save command
- ♦ Exit - execute the End command
- ♦ Cancel - execute the Cancel command (ignores all changes and re-displays the member selection list)

❒ The Edit pull-down has three choices

- ♦ Reset - execute a Reset command
- ♦ Undo - execute an Undo command
- ♦ Hilite... - display the Edit Color Settings pop-up window to set language-sensitive color editing, discussed next

Language-sensitive Color Editing

❒ **The ISPF Editor can set specific colors to language specific constructs, to help highlight the structure of your data**

Language constructs editor is sensitive to

- ♦ **Keywords**
- ♦ **Comments**
- ♦ **Quoted strings**
- ♦ **Compiler directives (only for C, COBOL, PL/I, PASCAL)**
- ♦ **User-selected special characters**
- ♦ **Program logic structures (logical blocks, IF/ELSE, parentheses matching)**
- ♦ **Strings matching FIND requests**
- ♦ **Miscellaneous features, depending on context**

Language-sensitive Color Editing, continued

Languages supported by the editor

- ♦ **Assembler**
- ♦ **BookMaster**
- ♦ **C**
- ♦ **COBOL**
- ♦ **Dialog Tag Language (DTL)**
- ♦ **HTML**
- ♦ **IDL - Interface Definition Language (for SOMobjects)**
- ♦ **ISPF panels (non-DTL)**
- ♦ **ISPF skeletons**
- ♦ **JCL**
- ♦ **Pascal**
- ♦ **REXX**
- ♦ **PL/I**
- ♦ **SuperC listing**
- ♦ **XML**
- ♦ **Other**

Language-sensitive Color Editing, continued

❒ The editor is not a language parser: it will not catch syntax errors or logic errors

- ♦ It simply provides the capability to highlight typical language structures, thus pointing out possible problems

❒ Highlighting is not available if the feature was not installed

❒ Highlighting is not available if records are greater than 255 bytes

❒ Highlighting may not work totally or correctly under an emulator

❒ Highlighting is not available when using formatted (DBCS) data

❒ You request language-sensitive color editing in one of two ways

- ♦ "Hilite..." choice from "Edit" pull-down from Edit / View panel Action Bar
- ♦ HILITE command being issued from the Command / Option line

Language-sensitive Color Editing, continued

❒ **The HILITE command (abbreviation: HI) specifies**

- **A coloring option (ON, OFF, LOGIC, IFLOGIC, DOLOGIC, NOLOGIC)**

- **A language (one of the supported languages or AUTO or DEFAULT)**
  - ✗ AUTO requests the editor to choose the language based on the first non-blank string it finds in the data

- **Various selection / processing options (RESET, PAREN, FIND, CURSOR, SEARCH, DISABLED)**

Syntax

```
HILITE   [ON | OFF | LOGIC | IFLOGIC | DOLOGIC | NOLOGIC]

    [AUTO | DEFAULT | OTHER | ASM | BOOK | C | COBOL | DTL |
    HTML | IDL | JCL | PANEL | PASCAL | PLI | REXX |
    SUPERC | SKEL | XML ]

    [RESET] [PAREN] [FIND] [CURSOR] [SEARCH] [DISABLED]
```

- **If you do not code any operands, you see a pop-up window for you to set values**

- **We will cover the various parameters then look at the pop-up**

Language-sensitive Color Editing, continued

❒ **HILITE command operands**

Coloring Options

♦ **IFLOGIC — turns on IF/ELSE logic matching; IF, ELSE, ENDIF and similar language-specific words display in the same color, different from the default color; unmatched ELSE keywords are highlighted in reverse video pink (!)**

♦ **DOLOGIC — turns on DO/END logic matching; DO, END and similar language-specific words display in the same color, different from the default color; unmatched END keywords are highlighted in reverse video pink**

✗ For BookMaster, match is :ol and :eol tags; for C, match is for "{" and "}" (C trigraphs for braces are not recognized)

♦ **LOGIC — turns on both IFLOGIC and DOLOGIC**

♦ **ON and NOLOGIC — both turn LOGIC coloring off and program coloring on**

♦ **OFF — turns coloring off except for CURSOR, FIND, and PAREN if specified**

❒ **Probably a good idea to first turn HI OFF and then set the values you want to use**

Language-sensitive Color Editing, continued

❒ **HILITE command operands, continued**

Language Options

- **AUTO - allow editor to determine the language**
- **DEFAULT - highlight data in a single color**
- **OTHER - highlight as a pseudo-PL/I style language; provides some support for, say, CLIST**
- **ASM - highlight data as Assembler**
- **BOOK - highlight data as BookMaster**
- **C - highlight data as C**
- **COBOL - highlight data as COBOL**
- **DTL - highlight data as Dialog Tag Language**
- **HTML - highlight data as HTML**
- **IDL - highlight data as Interface Definition Language**
- **JCL - highlight data as JCL**
- **PANEL - highlight data as Panel Language**
- **PASCAL - highlight data as Pascal**
- **PLI - highlight data as PL/I**
- **REXX - highlight data as Rexx**
- **SKEL - highlight data as ISPF skeleton language**
- **XML - highlight data as XML**

❒ **To "highlight data as** language**" means to use colors to show** language**-specific keywords, comments, quoted strings, and (C, COBOL, PL/I, Pascal only) compiler directives**

- **The user can also specify special characters to be highlighted**

Language-sensitive Color Editing, continued

❐ **HILITE command operands, continued**

Selection / processing options

- **FIND — when a Find command is issued, highlight all occurrences of the string in a user-specified color (default is reverse video white)**
  - ✗ Most, but not all, 'Find' features are supported (for example, Picture strings and labels are not supported)
  - ✗ HILITE FIND toggles FIND on and off

- **CURSOR — highlight entire word where cursor is positioned by highlighting from first non-blank to last non-blank characters in a user-specified color (default is white)**
  - ✗ HILITE CURSOR toggles CURSOR on and off

- **RESET — reset default HILITE settings (AUTO, ON, FIND, CURSOR)**

- **PAREN — turn on parentheses matching; all code is displayed in the default color; comments are displayed in a different color; unmatched parentheses are highlighted**

- **SEARCH — find first unmatched END, ELSE, } or) above the last displayed line on the screen, scroll so that line is at the top; may need to scroll to the bottom before issuing HILITE SEARCH**

- **DISABLED — turns off all HILITE features and removes all action bars**

Language-sensitive Color Editing, continued

❒ **Again, if you issue the HILITE command with no operands, you see the Edit Color Settings pop-up window to change settings through a dialog:**

```
 .------------------------------------------------------------------------------.
 |  File  Languages  Colors  Help                                               |
 | ---------------------------------------------------------------------------- |
 |                            Edit Color Settings                               |
 | Command ===>                                                                 |
 |                                                                              |
 |                                                              More      +     |
 | Language: 1_   1. Automatic      Coloring: 1  1. Do not color program        |
 |                2. Assembler                   2. Color program               |
 |                3. BookMaster                  3. Both IF and DO logic        |
 |                4. C                           4. DO logic only               |
 |                5. COBOL                       5. IF logic only               |
 |                6. HTML                                                       |
 |                7. IDL            Enter "/" to select option                  |
 |                8. ISPF DTL          Parentheses matching                     |
 |                9. ISPF Panel     _  Highlight FIND strings                   |
 |               10. ISPF Skeleton  _  Highlight cursor phrase                  |
 |               11. JCL            _                                           |
 |               12. Pascal         Note:  Information from this panel is       |
 |               13. PL/I           saved in the edit profile.                  |
 '------------------------------------------------------------------------------'
```

❒ **Clearly, you can set the current language, coloring strategy, and processing options (PAREN, FIND, CURSOR)**

- ♦ **Then, from the "File" Action Bar choice, you can choose:**

  - ✗ Restart application — apply these settings to all panels back to the point HILITE was invoked

  - ✗ Default All Settings — undo HILITE settings to settings existing before displaying this pop-up

  - ✗ Save and Exit — save HILITE settings and return to previous panel

  - ✗ Cancel — discard changes and return to previous panel

Language-sensitive Color Editing, continued

❒ **You can specify other changes by using other Action Bar choices**

- ♦ **<u>Help</u> - displays language-sensitive color editing related help topics**

- ♦ **<u>Colors</u> - provides three choices, each of which leads you to a pop-up window for setting colors used for special cases**

  - ✗ <u>Overtype Color...</u> — specify color to use for typed data

  - ✗ <u>Find String Color...</u> — specify color and highlighting used to indicate Find strings

  - ✗ <u>Cursor Phrase Color...</u> — specify color and highlighting used to highlight word / phrase the cursor is in

- ♦ **<u>Languages</u> - lets you display and change [some] settings for languages**

  - ✗ Provides a list of languages, or "All"; when you make a selection, you see the pop-up window shown on the next page

  - ✗ From this pop-up, you can change settings and / or go on to View the words designated as Keywords for this language

Language-sensitive Color Editing, continued

❒ **From the HILITE pop-up, use the "Languages" Action Bar choice and select a language; you see a pop-up like this:**

```
.------------------- Edit Color Settings --------------------.
|   File      View    Help                                    |
|----------------------------------------------------------- |
|          Language Element Color Specification               |
| Command===> ____________________________________________    |
|                                                             |
|     Language: COBOL                                         |
|                                                             |
|     Language Element        Color   Highlight               |
|     ---------------------- ------  ---------               |
|                                                             |
|     Default . . . . . . . . GREEN   NORMAL                  |
|     Comments  . . . . . . . TURQ    NORMAL                  |
|     Keywords  . . . . . . . RED     NORMAL                  |
|     Quoted Strings  . . . . WHITE   NORMAL                  |
|     Compiler Directives . . BLUE    NORMAL                  |
|     Special Characters  . . YELLOW  NORMAL                  |
|                                                             |
|     Special Characters to                                   |
|     Highlight  . . . . . . .                                |
|                                                             |
'-------------------------------------------------------------'
```

- **Again, you can change settings, then the "File" Action Bar choice gives you the same options as for the HILITE panel, and "Help" give you options related to the topic at hand**

- **The "View" Action Bar choice presents a pop-up that displays the keywords that are recognized for the current language**

  ✗ You are not allowed to change the list, it is just for reference

  ✗ Note that the person who supports ISPF can change the list of words by using IBM-supplied source code and Assembling and linking it; there is no support for adding languages

  ✗ The list for any given language may be out of date with later releases of the language if this portion of ISPF is not updated when you upgrade the compiler

MD Line Command

❒ MD - Make Dataline command

❒ Type command in on top of message, note, cols, or information lines (==MSG>, =NOTE=, =COLS>, or ======) in sequence number field, and line becomes part of the data

Example

```
000100 LIBERTY FLBJIBIT IS FLELLING AMONG
=COLS> ----+----1----+----2----+----3----+----
```

Type

```
MDOLS> ----+----1----+----2----+----3----+----
```

- And the columns line is made into a line of data

❒ Can also specify as:

```
MDn
```

- Or

```
MDMD
.
.
.
MDMD
```

VERSION and LEVEL Commands

When data is first created, if "STATS ON" is set in the profile, the data is given version and level values of 01 and 00 respectively

- ♦ **Version and level numbers are stored in the member statistics; level numbers may be stored in the data records also**

The editor normally increments the level number each time the data is changed (saved with modifications)

But you can change the version and level numbers through primary commands:

❒ **VERSION - change the version number of the data being edited**

```
COMMAND  ===>   VERSION 2
```

✗ **Sets the data being edited at version 2**

❒ **LEVEL - change the level number (within version) of the data being edited**

```
COMMAND  ===>  LEVEL 12
```

✗ **Sets the data being edited to modification level 12 (within the current version level)**

✗ **If issued before making changes, suppresses automatic incrementing of the level number**

Recent Edit / View Line Commands

❐ **The line commands introduced on these pages have recently been added to the edit and view capabilities; for z/OS V1.10:**

- **AK - "After and Keep": copy or move one or more lines after the line where this command is found, but keep the content of the copy or move to repeat it in a series of copies or moves**
- **BK - "Before and Keep": copy or move one or more lines before the line where this command is found, but keep the content of the copy or move to repeat it in a series of copies or moves**
- **OK - "Overlay and Keep": copy or move one or more lines on top of the line(s) where this command is found, but keep the content of the copy or move to repeat it in a series of copies or moves**

<u>Syntax</u>

AK	place data after this line (target for M or C):	AK	AK*n*
BK	place data before this line (target for M or C):	BK	BK*n*
OK	place data over this line (target for M or C):	OK	OK*n* OOK - - -OOK

- **The editor starts at the highest M or C line command and grabs the line(s) indicated, then moves to the top and proceeds down the file, looking for targets**
- **For each AK, the grabbed lines are copied after that line**
- **For each BK, the grabbed lines are copied before that line**
- **For each OK, the grabbed lines overlay the target**
- **There must be a B, an A, or an O command to end the series, thus freeing the grabbed lines**
  - ✗ And deleting from their original spot them if the initial command was M

Recent Edit / View Line Commands, 2

Example

- Suppose you have many lines of code all jumbled together and you'd like to insert some blank or comment lines, for readability:

```
000019        Data division.
000020        File section.
000021        FD  parts.
000022        01  wk-rec.
000023            05  wk-partno             pic x(9).
000024            05  wk-descrption         pic x(30).
000025            05                        pic x(5).
000026            05  wk-unit-price         pic s9999v999 packed-decimal.
000027            05  wk-qty-hand           pic s9(5)     packed-decimal.
000028            05                        pic x.
000029            05  wk-qty-order          pic s9999     binary.
000030            05  wk-reorder            pic s9999     binary.
000031            05                        pic x(11).
000032            05  wk-category           pic x(10).
000033            05                        pic x(23).
000034        Working-storage section.
000035        01  blank-line pic x value x'15'.
000036       * data items for environment variable work  ------------
000037        01  Envar-related-variables.
000038            02  var-name      pic x(13)   value z'QUERY_STRING'.
000039            02  env-ptr       pointer.
000040            02  err-ind       redefines env-ptr pic s9(9) binary.
000041            02  string-len    pic s9(8)   binary value 0.
000042            02  partno        pic x(9)           value spaces.
000043            02  unpr          pic 9999.999.
000044            02  quantity      pic 99999.
000045       * items used in VSAM alloation and processing  ------
000046        01  bpxwdyn           pic  x(8)  value 'BPXWDYN'.
000047        01  allocate-request.
000048            02                pic s9(4)  binary value 50.
000049            02                pic x(50)  value
000050                   'alloc fi(parts) dsn(scomsto.train.ksds) shr '.
000051        01  free-request.
000052            02                pic s9(4)  binary value 24.
000053            02                pic x(24)  value 'free fi(parts) '.
000054        01  vsam-stat  pic 99.
```

- First, insert a blank line after line 19 (describe the process), resulting in:

Recent Edit / View Line Commands, 3

Example, continued

```
000019        Data division.
000020
000021        File section.
000022        FD  parts.
000023        01  wk-rec.
000024            05  wk-partno             pic x(9).
000025            05  wk-descrption         pic x(30).
000026            05                        pic x(5).
000027            05  wk-unit-price         pic s9999v999 packed-decimal.
000028            05  wk-qty-hand           pic s9(5)     packed-decimal.
000029            05                        pic x.
000030            05  wk-qty-order          pic s9999     binary.
000031            05  wk-reorder            pic s9999     binary.
000032            05                        pic x(11).
000033            05  wk-category           pic x(10).
000034            05                        pic x(23).
000035        Working-storage section.
000036        01  blank-line pic x value x'15'.
000037       * data items for environment variable work  ------------
000038        01  Envar-related-variables.
000039            02  var-name      pic x(13)   value z'QUERY_STRING'.
000040            02  env-ptr       pointer.
000041            02  err-ind       redefines env-ptr pic s9(9) binary.
000042            02  string-len    pic s9(8)   binary value 0.
000043            02  partno        pic x(9)           value spaces.
000044            02  unpr          pic 9999.999.
000045            02  quantity      pic 99999.
000046       * items used in VSAM alloation and processing  ------
000047        01  bpxwdyn           pic  x(8)  value 'BPXWDYN'.
000048        01  allocate-request.
000049            02                pic s9(4)  binary value 50.
000050            02                pic x(50)  value
000051                   'alloc fi(parts) dsn(scomsto.train.ksds) shr '.
000052        01  free-request.
000053            02                pic s9(4)  binary value 24.
000054            02                pic x(24)  value 'free fi(parts) '.
000055        01  vsam-stat  pic 99.
```

- ♦ **Now we want to copy that blank line to several places, to provide more visual separation, perhaps like this:**

Recent Edit / View Line Commands, 4

Example, continued: before:

```
000019        Data division.
c00020
ak0021        File section.
000022        FD  parts.
000023        01  wk-rec.
000024            05  wk-partno             pic x(9).
000025            05  wk-descrption         pic x(30).
000026            05                        pic x(5).
000027            05  wk-unit-price         pic s9999v999 packed-decimal.
000028            05  wk-qty-hand           pic s9(5)     packed-decimal.
000029            05                        pic x.
000030            05  wk-qty-order          pic s9999     binary.
000031            05  wk-reorder            pic s9999     binary.
000032            05                        pic x(11).
000033            05  wk-category           pic x(10).
ak2 34            05                        pic x(23).
ak0035        Working-storage section.
ak0036        01  blank-line pic x value x'15'.
ak0037       * data items for environment variable work  ------------
000038        01  Envar-related-variables.
000039            02  var-name      pic x(13)   value z'QUERY_STRING'.
000040            02  env-ptr       pointer.
000041            02  err-ind       redefines env-ptr pic s9(9) binary.
000042            02  string-len    pic s9(8)   binary value 0.
000043            02  partno        pic x(9)          value spaces.
000044            02  unpr          pic 9999.999.
ak2 45            02  quantity      pic 99999.
000046       * items used in VSAM alloation and processing  ------
ak0047        01  bpxwdyn       pic  x(8)  value 'BPXWDYN'.
000048        01  allocate-request.
000049            02                pic s9(4)  binary value 50.
000050            02                pic x(50)  value
ak0051                   'alloc fi(parts) dsn(scomsto.train.ksds) shr '.
000052        01  free-request.
000053            02                pic s9(4)  binary value 24.
0a0054            02                pic x(24)  value 'free fi(parts) '.
000055        01  vsam-stat  pic 99.
```

- **Then press <Enter> and we see it's much easier to read ...**

Recent Edit / View Line Commands, 5

Example, continued: after:

```
000019        Data division.
000020
000021        File section.
000022
000023        FD  parts.
000024        01  wk-rec.
000025            05  wk-partno             pic x(9).
000026            05  wk-descrption         pic x(30).
000027            05                        pic x(5).
000028            05  wk-unit-price         pic s9999v999 packed-decimal.
000029            05  wk-qty-hand           pic s9(5)     packed-decimal.
000030            05                        pic x.
000031            05  wk-qty-order          pic s9999     binary.
000032            05  wk-reorder            pic s9999     binary.
000033            05                        pic x(11).
000034            05  wk-category           pic x(10).
000035            05                        pic x(23).
000036
000037
000038        Working-storage section.
000039
000040        01  blank-line pic x value x'15'.
000041
000042       * data items for environment variable work  ------------
000043
000044        01  Envar-related-variables.
000045            02  var-name      pic x(13)   value z'QUERY_STRING'.
000046            02  env-ptr       pointer.
000047            02  err-ind       redefines env-ptr pic s9(9) binary.
000048            02  string-len    pic s9(8)   binary value 0.
000049            02  partno        pic x(9)           value spaces.
000050            02  unpr          pic 9999.999.
000051            02  quantity      pic 99999.
000052
000053
000054       * items used in VSAM alloation and processing  ------
000055        01  bpxwdyn           pic  x(8)  value 'BPXWDYN'.
000056
000057        01  allocate-request.
000058            02                pic s9(4)  binary value 50.
000059            02                pic x(50)  value
000060                   'alloc fi(parts) dsn(scomsto.train.ksds) shr '.
000061
000062        01  free-request.
000063            02                pic s9(4)  binary value 24.
000064            02                pic x(24)  value 'free fi(parts) '.
000065
000066        01  vsam-stat  pic 99.
```

Recent Edit / View Line Commands, 6

❒ **The line commands introduced on these pages have recently been added to the edit and view capabilities; for z/OS V1.11:**

- **HX - Display one or more lines in hexadecimal, leaving the other lines alone**

Syntax

HX   display in hex: HX   HX*n*   HXX - - - HXX

Example - using a few lines from our previous example

```
000023        FD  parts.
hx0024        01  wk-rec.
000025            05  wk-partno             pic x(9).
```

- **yields:**

```
000023        FD  parts.
000024        01  wk-rec.
       4444444FF44A96988444444444444444444444444444444444444444444444444
       000000001006209530000000000000000000000000000000000000000000000000
000025            05  wk-partno             pic x(9).
```

- **Note that the primary command HEX OFF will reset any and all lines displayed in hex in the current file**

Computer Exercise: Using Tabs and other Profile Characteristics

Using EDIT, accomplish the following tasks

1. Edit a new member called PDFDATA2
   - a. Examine your current edit profile setting; make any changes you feel necessary
   - b. Set hardware tabs at locations 11, 39, and 51
   - c. Establish the dollar sign, $, as your logical tab character and enter the following lines:

```
EMPLOYEE$LAST$FIRST
NUMBER$NAME$NAME$DEPT.

02027$THORNAPPLE$SEBASTIAN$103
02037$CRABAPPLE$THORNY$103
02047$PINEAPPLE$TIMOTHY$103
02057$WOODS$ARNOLD$105
07043$FAIRPLAY$$105
09017$STORM$HELEN$107
```

- remember to allow for the blank line.

   - d. Change all occurrences of APPLE to LEMON
   - e. Sort the detail lines (lines with five numeric digits in first five columns) by last name
   - f. Create a new member called DETAILS consisting of only the detail lines of this data

2. Save your current profile under the name of CPROF1.

3. Create a new profile called CPROF2 with the following attributes: CAPS OFF, NUM OFF, and TABS OFF.

4. Experiment with the Language-sensitive Color Editing facility, if it is available.

   In the libraries listed below, there are members written in the languages mentioned; copy some of these into your LIBRARY PDS and experiment with the HILITE-ing.

   Remove an ELSE from an IF construct (C, COBOL, PL/I only) and experiment with IFLOGIC; remove a closing parenthesis and experiment with SEARCH.

   Experiment changing color assignments; view the reserved word list for some language.

Library-------------------------------------	Member---	Written in
<userid>.TRAIN.LIBRARY	TRF2F	Assembler
_________.TRAIN.LIBRARY	LEC	C
	LECOBOL	COBOL
	LEPLI	PL/I
	JONION	JCL
	STARTER	REXX
	SETSTOP	ISPF panel language
	CCATLOG	ISPF skeleton language

Section Preview

❒ **Data Set List Utility and Commands**

- ♦ **Option 3.4: Data Set List Does It All**
- ♦ **Commands**
- ♦ **DSLIST and Commands (Machine Exercise)**

Option 3.4: Data Set List Does It All

❒ **Versatile utility that allows you to:**

- ♦ **Print or display VTOC information about a volume**
- ♦ **Print a list of data set names**
- ♦ **Display a list of data set names**
  - ✗ **And here is where the power comes in ...**

❒ **The lines in this list of data set names can have line commands issued against one or more of them**

- ♦ **These line commands accomplish many functions in other ISPF/PDF options that we have already examined ...**

DSLIST, 2

❒ **Once you have a list of data set names on display, you can issue the following line commands against any name in the display:**

Line Command	Facility
V	View data set; same as Option 1
B	Browse data set: same as Browse command under Edit / View
E	Edit data set: same as Option 2
*D	Delete data set: same as **D** Suboption of 3.2
*DEL	Delete data set
R	Rename data set: same as **R** Suboption of 3.2
I	Data set information: same as ' ' Suboption of Option 3.2
S	Information (Short): same as **S** Suboption of Option 3.2
C	Catalog data set: same as **C** Suboption of Option 3.2
U	Uncatalog data set: same as **U** Suboption of Option 3.2
P	Print entire data set: same as **L** Suboption of Option 3.1
PX	Print index (directory) listing: same as **X** Suboption of Option 3.1

Notes

- **D causes DSLIST to issue SVC99**
- **DEL causes ISPF to invoke IDCAMS**
  - ✗ Can delete catalog aliases and GDG bases (**D** can not)
  - ✗ If select an uncataloged data set, if there is a cataloged data set with the same name, the cataloged data set gets deleted!

DSLIST, 3

❒ **Once you have a list of data set names on display, you can issue the following line commands against any name in the display:**

Line Command	Facility
M	Display member list: list similar to Option 3.1 • But line command room is larger • Can issue line commands against member names in this list • Including **E** to edit, **B** to browse, **D** to delete, **R** to rename, **P** to print
Z	Compress PDS: same as **C** Suboption of Option 3.1
F	Free unused disk space in data set (unique to 3.4)

✗ For example, if 30 cylinders had been allocated, but only 12 cylinders were being used, 18 cylinders would be freed up for use by other data sets

RS	Reset or delete ISPF statistics: same as **G** option of 3.1
MO	Move: same as **M** under 3.3
CO	Copy: same as **C** under 3.3
RA	RefAdd command to add data set to personal data set list; may use **A** for abbreviation
X	eXclude line from the displayed list
NX	uNeXclude line(s) from the displayed list
NXF	uNeXclude First data set(s) from the displayed list
NXL	uNeXclude Last data set(s) from the displayed list

DSLIST, 4

❒ The initial panel for DSLIST looks like this:

```
  Menu  Reflist  RefMode  Utilities  Help
----------------------------------------------------------------------------
                            Data Set List Utility
Command ===> ___________________________________________________

  blank Display data set list           P Print data set list
      V Display VTOC information       PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . .  dept53
   Volume . . . . . .  ______

Data set list options:
   Initial view . . 1  1. Volume             Enter "/" to select option
                       2. Space              /  Confirm Delete
                       3. Attrib             /  Confirm Member Delete
                       4. Total              /  Include Additional Qualifiers
                                             /  Display Catalog Name
                                             /  Display Total Tracks
                                             _  Prefix Dsname level

When the data set list is displayed, enter either:
  "/" on the data set list command field for command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
  "=" to execute the previous command.
```

Notes

- **When you use a referral list from this utility, before the data set name is filled in:**
  - ✗ Quotes are removed from the data set name
  - ✗ The current PREFIX value is prefixed to the name, unless it is already the high level prefix
  - ✗ Any member name is removed

- **If you select Confirm Delete or Confirm Member Delete, whenever you try to delete a data set or a member from a library you will see the appropriate Confirm Delete pop-up**
- **If you do not select Include Additional Qualifiers, the dsname level field is considered to reflect all the qualifiers you want**
- **If you select Display Catalog Name, that will show in the Total view**
- **Selecting Display Total Tracks will give you the number of all tracks in all files in the list (ISPF 5.9)**

DSLIST, 5

❒ **You request a data set list display by entering either a high level qualifier for data sets, or a volume serial to be searched (or both)**

Examples

```
Dsname Level . .  dept53
Volume . . . . .  ______
```

♦ **Will produce a list of all disk data sets with a high level qualifier of DEPT53** (if Include Additional Qualifiers selected; otherwise, just any data set with name DEPT53)

```
Dsname Level . .  dept53.train
Volume . . . . .  dopey
```

♦ **Will produce a list of all disk data sets with a high level qualifier of DEPT53.TRAIN on the volume with volume serial DOPEY** (if Include Additional Qualifiers selected; otherwise, just data sets with name DEPT53.TRAIN)

```
Dsname Level . .  dept53.tr*
Volume . . . . .  ______
```

♦ **Will produce a list of all disk data sets with a high level qualifier of DEPT53, a second level qualifier beginning with the letters TR and any other levels to the name** (if Include Additional Qualifiers selected; otherwise, just data set names with two levels, the first DEPT53, the second beginning with TR)

♦ **As of z/OS 1.11, if Prefix Dsname level is selected, your current TSO prefix is prepended to the Dsname Level value, unless that value is in quotes**

DSLIST, 6

❒ **Use wildcard characters in Dsname Level as follows:**

- ♦ **A percent sign (%) indicates a single "don't care" character (***e.g.***: dept53.new0%t)**

- ♦ **An asterisk (*) within a qualifier indicates zero or more "don't care" characters (***e.g.***: stnt329.tr*s%t)**

- ♦ **An asterisk (*) by itself indicates one or more levels may be specified (***e.g.***: pshcom.*.cobol)**

- ♦ **A double asterisk (**) by itself indicates zero or more levels (***e.g.***: $trncm.**.cntl)**

❒ **Normally, you fully qualify the first level**

- ♦ **However, you may use wildcards in the first level qualifier also (***e.g.***: **.cobol), although some shops prohibit this**

Caution

- ♦ **If you enter a high-level qualifier of * or **, ISPF shows a pop-up window warning you that the search will be on all catalogs on the system and may take some time**

  ✗ At this point, you can cancel or continue the process

- ♦ **Also, at least one qualifier must be partially qualified (***e.g.***: *.* is not supported)**

- ♦ **Also note the impact of the Prefix Dsname level entry described on the previous page**

DSLIST Views

❐ **The list produced by a DSLIST request may be formatted in one of four ways, called views:**

Volume view

✗ **Lists data set names and the volume serial on which they reside**

Space view

✗ **Lists data set names and number of tracks allocated and used, and the device type**

Attrib view

✗ **Lists data set names, data set organization, recording format, record size, and block size**

Total view

✗ **Lists all of the above plus the creation, expiration, and last referenced dates**

✗ **Requires two lines per entry**

❐ **In z/OS 1.9 and later, if you selected "Display Total Tracks" the Space view and Total view will have an extra line with this information**

DSLIST Views, 2

❒ **You specify which view you want first**

- ♦ **The alternative views are set in a 'ring' or circle: as you scroll left or right (F10 & F11), you get the next view in the circle, eventually going back around to the original view**

❒ **DSLIST restriction: no support for tape (until z/OS 1.8)**

DSLIST Primary Commands

❒ **From each of the various views, you can issue primary commands as follows:**

APPEND [list_name | dsname_level]	—	Add more names to the list (if omit parameter, get pop-up of personal list)
CONFIRM [ON|OFF]	—	Change request for confirm-on-delete pop-up
EXCLUDE	—	Exclude lines from list containing a specified string (use Edit/View options)
FIND	—	Search Dsname column for character string
RFIND	—	Repeat last find
REFRESH	—	Refresh list
RESET	—	Unexclude excluded lines, remove pending line commands
SORT	—	Sort list on data set name or attribute
LOCATE	—	Search for value in field data set list is currently sorted on
LC	—	Bring up the DSLIST Color change utility panel
SAVE	—	Save list to a data set for later review or processing
VA, VS, VT, VV	—	Change the view to Attribute, Space, Total, Volume, respectively
MEMBER	—	Search all libraries in the list for members with names that match a search string
SRCHFOR	—	search all files for a string in the data (details beyond the scope of this course)

DSLIST Primary Commands, continued

❒ **The MEMBER primary command is fairly new, and can be quite handy**

Syntax

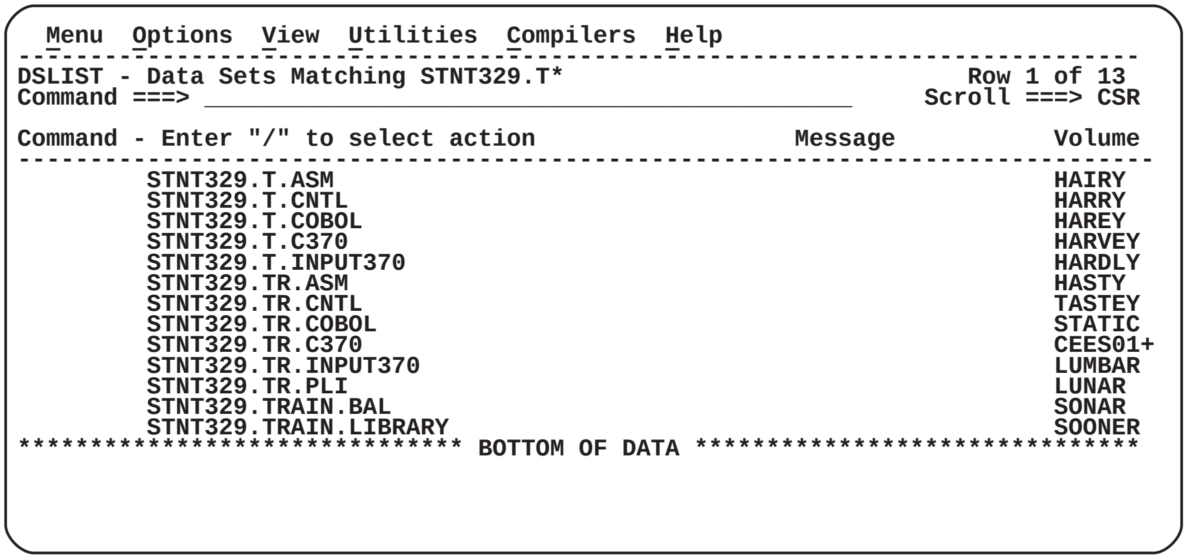
MEMBER pattern **[{X | EX} | NX] [RECALL1 | RECALL2]**

Where

- ✗ MEMBER may be abbreviated MEM or even M
- ✗ *pattern* is any member name pattern as previously discussed
- ✗ By default, search all libraries in the list, except for HSM migrated libaries
- ✗ X (or EX) only looks in libraries that are excluded from the list
- ✗ NX only looks in libraries not excluded from the list
- ✗ RECALL1 says also look in libraries that are HSM migrated to DASD
- ✗ RECALL2 says also look in libraries that are HSM migrated to DASD or TAPE

Data Set List Example

❒ **Once you have specified Dsname Level and / or Volume information, you get a list of data sets that satisfy the list criteria:**



```
  Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
DSLIST - Data Sets Matching STNT329.T*                              Row 1 of 13
Command ===> ____________________________________________       Scroll ===> CSR

Command - Enter "/" to select action                  Message           Volume
-------------------------------------------------------------------------------
         STNT329.T.ASM                                                  HAIRY
         STNT329.T.CNTL                                                 HARRY
         STNT329.T.COBOL                                                HAREY
         STNT329.T.C370                                                 HARVEY
         STNT329.T.INPUT370                                             HARDLY
         STNT329.TR.ASM                                                 HASTY
         STNT329.TR.CNTL                                                TASTEY
         STNT329.TR.COBOL                                               STATIC
         STNT329.TR.C370                                                CEES01+
         STNT329.TR.INPUT370                                            LUMBAR
         STNT329.TR.PLI                                                 LUNAR
         STNT329.TRAIN.BAL                                              SONAR
         STNT329.TRAIN.LIBRARY                                          SOONER
******************************* BOTTOM OF DATA *******************************
```

❒ **Then enter line commands next to the data set name**

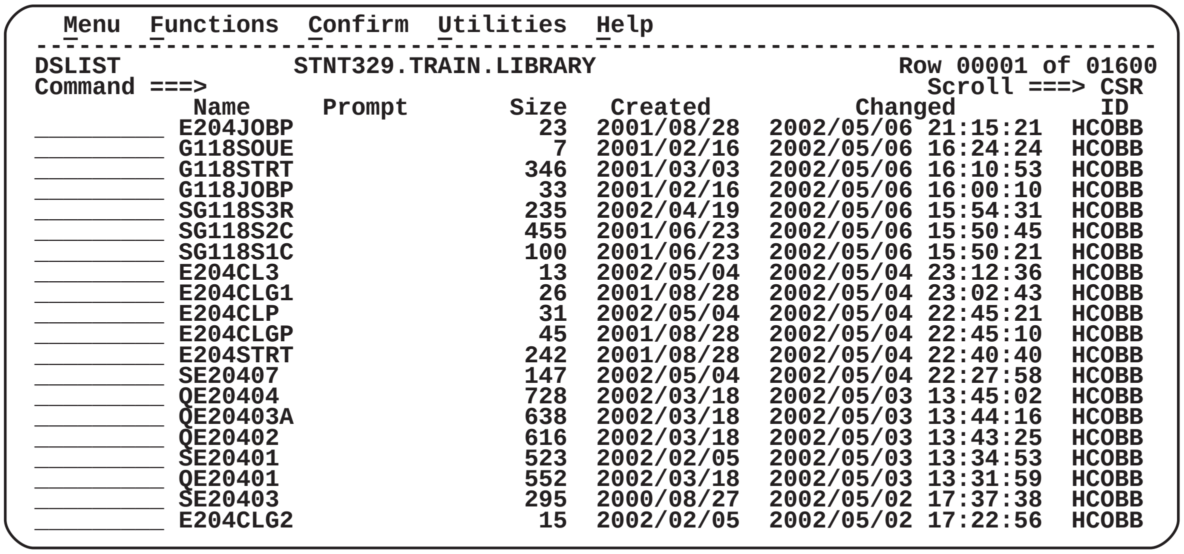
- ♦ **The Command field and the field containing the data set name together make up a single point-and-shoot field**
- ♦ **If you select a line and press <Enter> you'll see a pop-up with the list of standard commands available**

❒ **For Libraries (PDS/PDSE), the line commands V, B, E, and M produce a member list display**

- ♦ **Each member will have a 9-character comand field so you can enter commands such as: V, B, E, R, EX, P and more**

Member List Example

❒ **Here is an example of a panel after requesting option M, for Memberlist:**



```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
DSLIST           STNT329.TRAIN.LIBRARY                    Row 00001 of 01600
Command ===>                                                 Scroll ===> CSR
           Name     Prompt       Size   Created          Changed          ID
_________ E204JOBP                 23  2001/08/28  2002/05/06 21:15:21  HCOBB
_________ G118SOUE                  7  2001/02/16  2002/05/06 16:24:24  HCOBB
_________ G118STRT                346  2001/03/03  2002/05/06 16:10:53  HCOBB
_________ G118JOBP                 33  2001/02/16  2002/05/06 16:00:10  HCOBB
_________ SG118S3R                235  2002/04/19  2002/05/06 15:54:31  HCOBB
_________ SG118S2C                455  2001/06/23  2002/05/06 15:50:45  HCOBB
_________ SG118S1C                100  2001/06/23  2002/05/06 15:50:21  HCOBB
_________ E204CL3                  13  2002/05/04  2002/05/04 23:12:36  HCOBB
_________ E204CLG1                 26  2001/08/28  2002/05/04 23:02:43  HCOBB
_________ E204CLP                  31  2002/05/04  2002/05/04 22:45:21  HCOBB
_________ E204CLGP                 45  2001/08/28  2002/05/04 22:45:10  HCOBB
_________ E204STRT                242  2001/08/28  2002/05/04 22:40:40  HCOBB
_________ SE20407                 147  2002/05/04  2002/05/04 22:27:58  HCOBB
_________ QE20404                 728  2002/03/18  2002/05/03 13:45:02  HCOBB
_________ QE20403A                638  2002/03/18  2002/05/03 13:44:16  HCOBB
_________ QE20402                 616  2002/03/18  2002/05/03 13:43:25  HCOBB
_________ SE20401                 523  2002/02/05  2002/05/03 13:34:53  HCOBB
_________ QE20401                 552  2002/03/18  2002/05/03 13:31:59  HCOBB
_________ SE20403                 295  2000/08/27  2002/05/02 17:37:38  HCOBB
_________ E204CLG2                 15  2002/02/05  2002/05/02 17:22:56  HCOBB
```

♦ **Note that in this example, the member list has been sorted by "Changed": descending sequence by the date and time in the Changed column**

Some of the available line commands for member lists

B	**browse**
C	**copy**
D	**delete**
E	**edit**
G	**reset statistics**
J	**submit as a job**
M	**move**
P	**print**
R	**rename**
T	**TSO command**
V	**view**
W	**workstation command**

What You Can't Do Under DSLIST

Allocate new files

- **Until z/OS 1.13; then you can issue a line command AL to allocate a file like the file on that line; *e.g.*, if you start with:**

```
             STNT329.T.COBOL                    HAREY
```

... then type over that line:

```
   al  test.cobol329.T.COBOL                    HAREY
```

  - ✗ After you press <Enter> you will see a dialog for finishing the allocation of, in this case, <*userid*>.TEST.COBOL

Edit a member that does not exist from the MEMBERLIST option

- **Although you can E (edit) a member that does exist, then create a new member by issuing "EDIT *newmembername*"**

  - ✗ Using edit-under-edit to create the new member

  - ✗ This still will not work if there are no members in the library to start with

- **Similarly, you can select a library using E (Edit) and then you can create a new member from the member list by issuing the Select primary command for a member that doesn't exist:**

```
===> s job212
```

  - ✗ You are placed in edit of the new, empty, member

Commands

❒ **We have already encountered a large number of ISPF commands**

- ♦ **There are more, but we are almost done with the ones we are going to cover in this course**

❒ **In addition, you can issue commands to TSO**

❒ **At this point, we examine how to pass commands to TSO**

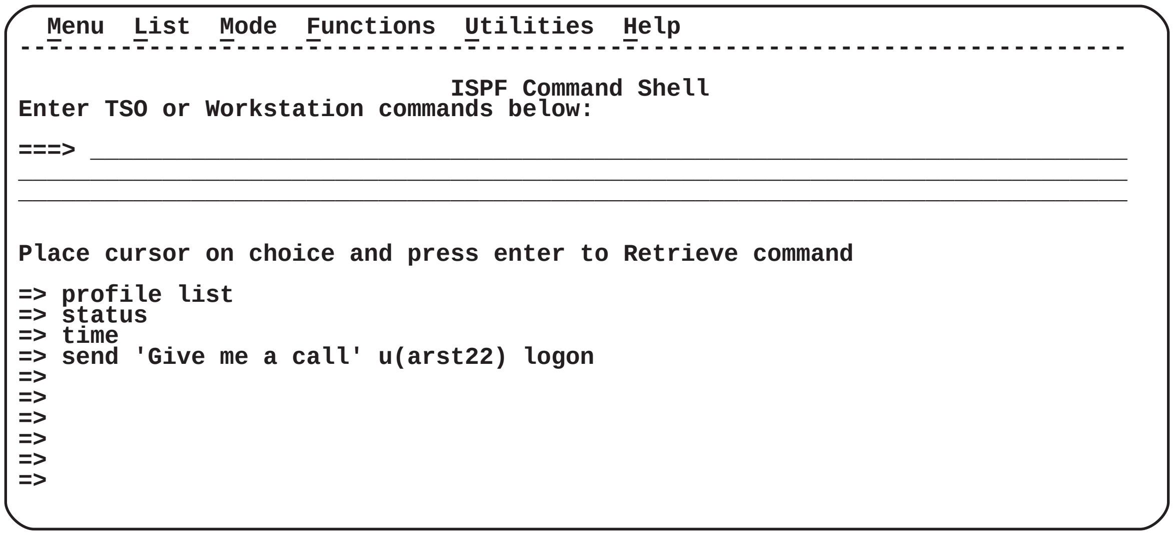
❒ **Command Interfaces**

- ♦ **To issue a command to TSO, from any Command / Option line enter "TSO" followed by the TSO command**

- ♦ **Alternatively, select Option 6 from the Primary Option Menu, or option 5 from the Menu Action Bar choice: the ISPF Command Shell dialog ...**

The ISPF Command Shell

❒ **The Command Shell panel looks like this**



```
  Menu  List  Mode  Functions  Utilities  Help
-------------------------------------------------------------------------------

                              ISPF Command Shell
Enter TSO or Workstation commands below:

===> _______________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________


Place cursor on choice and press enter to Retrieve command

=> profile list
=> status
=> time
=> send 'Give me a call' u(arst22) logon
=>
=>
=>
=>
=>
=>
```

- **The panel saves up to 10 commands that you have saved**
  - ✗ Either using automatic save or explicitly saving them
- **These are point-and-shoot fields: select a line and press <Enter> and the command is processed:**
  - ✗ Copied into the command area if your current Mode (set from the Action Bar) is Retrieve
  - ✗ Copied and executed if your current Mode is Execute
  - ✗ Deleted from the list, without being executed, if your current Mode is Delete

The ISPF Command Shell, continued

❒ **The other Action Bar choices are typical**

- ♦ **<u>Menu</u> provides the standard list of options**

- ♦ **<u>List</u> sets Update On (each command you enter is automatically added to the list) or Update Off (commands are not automatically added to the list)**

- ♦ **<u>Mode</u> sets the Mode as Retrieve, Execute, or Delete**

- ♦ **<u>Functions</u> provides these choices**

  - ✗ <u>Compress list</u> - remove duplicate commands and extra spaces from the saved command area (only needed if Update Off; compress is automatic if Update is On)

  - ✗ <u>TSO command</u> - sets shell to route commands to TSO

- ♦ **<u>Utilities</u> is the standard list, and <u>Help</u> is a list of related topics**

Sample Commands

❐ **There are lots of commands available**

- ♦ **Details are in different courses, but here are a few commands to play with ...**

- ♦ **The commands shown in the command save area on page 313 are all valid TSO commands**

    - ✗ The **send** command includes a message text (bounded in single quotes), a list of 1 to 20 TSO id-s the message goes to (in parentheses, as part of the **u** parameter (for "user"), separated by commas and / or spaces), and possibly the word **logon** to say "if a user is not logged on now, save the message until they logon again"

Edit Line Commands

- ❐ **In edit, you can assign line commands to function keys**

  - ♦ **However, commands assigned to function keys are normally executed as if they are primary commands**

  - ♦ **If you prefix the command with a colon (:), then the editor strips off the colon and issues the line command as if it had been entered in the sequence number field of the line where the cursor is**

  - ♦ **If the cursor is not in a data line, the command is ignored**

  <u>For example</u>

  - ♦ **If you have these key settings:**

    PF15 . . . :lc
    PF16 . . . :ts

  - ♦ **Then putting the cursor on a line and pressing function key 15 lower cases the data on that line**

  - ♦ **Putting the cursor on a line and pressing function key 16 causes a text split on the line where the cursor is at, at the position where the cursor is at**

- ❐ **Very nice for "ts", "hx", and several other line commands**

Computer Exercise: DSLIST and Commands

1. Use option 3.4 to accomplish the following:
   a. Create a list of all data sets beginning with your high level qualifier.
   b. Examine all the views of this list
   c. Obtain a member list of <userid>.TRAIN.LIBRARY
   d. Delete the members named BASE, ARAGON, DATAREAS, FILES, INITIALZ, LOGIC, and TERMINAT
   e. Browse the member named OLDPOEM

2. Under ISPF 6, issue the TSO TIME command; use the TSO SEND command to SEND a message to one other student in the class (see page 315 for information on syntax); examine the output from STATUS and PROFILE LIST commands

This page intentionally left almost blank.

Section Preview

❒ **Introduction to JCL**

- ♦ **Operating System**
- ♦ **The Application Program Environment**
- ♦ **MVS - Multiple Virtual Storages**
- ♦ **The Road to z/OS**
- ♦ **z/OS Work Flow**
- ♦ **Job Entry Subsystem**
- ♦ **JCL Statement Format**
- ♦ **JOB Statement Format**
- ♦ **EXEC Statement Format**
- ♦ **OSWTO (Machine Exercise)**
- ♦ **JCL Clues - 1**

Operating System

❐ **A Collection of programs that:**

- ♦ **Manage a computer system's resources**

  - **Maximize device utilization**
  - **Transfer data between memory and devices at program request**
  - **Handle error detection and recovery**
  - **Attain maximum possible performance under current workload**

- ♦ **Schedule work to be done**

  - **Determine jobs to be run, based on job control statements**
  - **Assign (allocate) resources to programs as necessary**
  - **Handle unscheduled work such as time sharing systems and transaction processing work**
  - **Communicate with operator <u>via</u>**
    <u>Commands</u> (operator to system)
    <u>Messages</u> (system to operator)

- ♦ **Maintain integrity of system and data**

  - **Provide security**
  - **Prevent simultaneous update**
  - **Prevent deadlock**

The Application Program Environment

- ❒ **An operating system provides an environment, a context, for application programs to run**
  - ♦ **Control blocks keep track of all programs in memory, their location, attributes, and status**
  - ♦ **System services allow application programs to do I/O, manage memory dynamically, handle application errors, and much more**

- ❒ **The most meaningful perspective here is how memory is organized, and we explore this in the following pages**
  - ♦ **To show how memory is organized and, briefly, why it is organized that way**
  - ♦ **We do this by looking at a short history of MVS, MVS/XA, OS/390, then z/OS**
    - ✗ Roughly corresponding to addressing limits of 24-bits (MVS), 31-bits (MVS/XA and OS/390), and 64-bits (z/OS): the size of memory the hardware and software support

MVS - <u>M</u>ultiple <u>V</u>irtual <u>S</u>torages

❒ **An operating system that runs on S/370 and later IBM mainframes**

- **<u>Virtual Storage</u> - the functional illusion of computer internal memory (storage), created using real internal memory, disk as a backing store, and hardware features of the CPU to map virtual addresses to real addresses**

  - ✗ Only the portions of virtual memory holding data and instructions currently being used need to be in real memory at any point in time

- **<u>Address Space</u> - a virtual storage that appears to be as large as the hardware addressing scheme allows (24-bit addresses, which allow up to 16MB of virtual storage per address space)**

  - ✗ Contains operating system code and user code
  - ✗ Contains data currently being processed

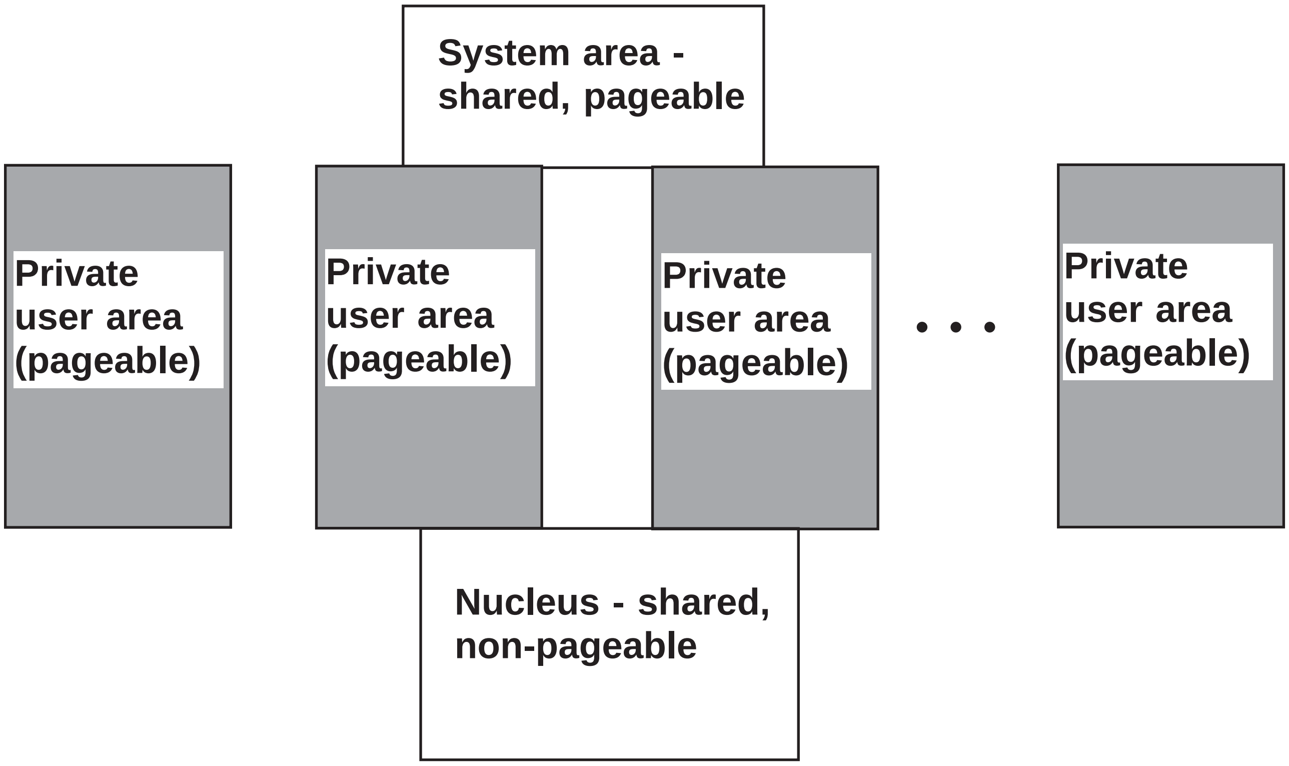
- **Each user has their own, distinct Address Space**

  - ✗ System Address Spaces
  - ✗ Batch Jobs
  - ✗ Time Sharing Users
  - ✗ Maximum of 32,767 Address Spaces total

- Because each user has their own address space, each address space needs to have a copy of the operating system

  - Since this is the same for each user, the addressing scheme is set up to have only one, shared, copy of the nucleus area and the system area

  - The unique parts of an MVS system look, conceptually, like this:


System area - shared, pageable
Private user area (pageable)
Private user area (pageable)
Private user area (pageable)
. . .
Private user area (pageable)
Nucleus - shared, non-pageable

- Again, addresses are 24-bits so each address space is 16MB in size

The Road to z/OS, 2

- ❐ **In the 1980's, IBM bit the bullet and extended the address space from 24 bits to 31 bits**
  - ♦ **31 bits instead of 32 bits for a variety of reasons, which provides for a 2 GB address space (2,147,483,648 bytes)**
  - ♦ **This was called extended architecture, abbreviated XA, so the operating system was called MVS/XA**
  - ♦ **This provides for 128 times the previous amount of virtual storage for programs to use**
  - ♦ **In addition to providing a larger address space, IBM re-arranged the layout**
    - ✗ Sections of code that relied on 24-bit addresses had to remain under the 16 MB limit (which has come to be called The Line)
    - ✗ So IBM moved as much of their code as possible above The Line (there will always have to be some code below The Line, to support older code)
  - ♦ **So, the layout of an address space in MVS/XA looks like the diagram on the following page ...**

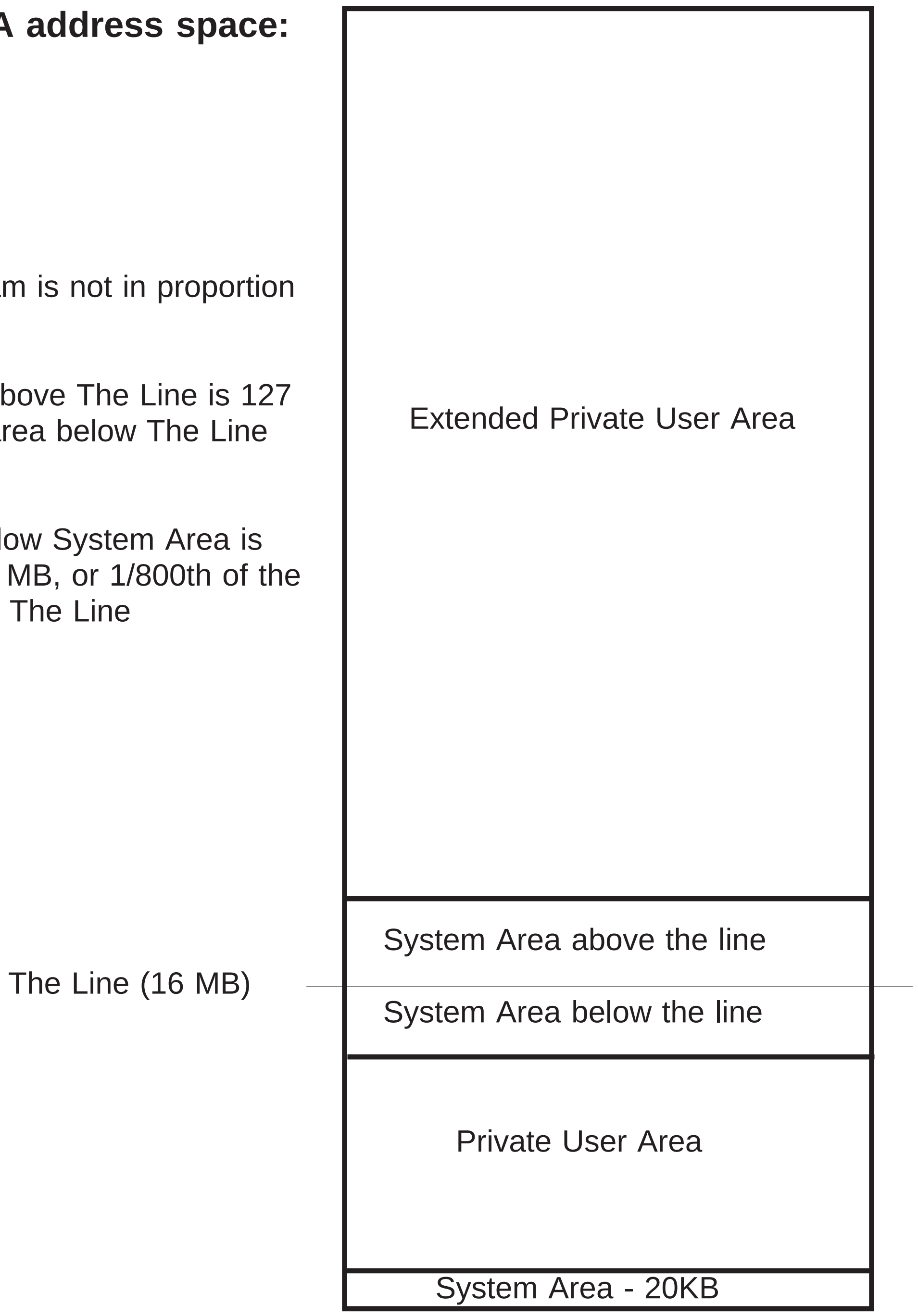
The Road to z/OS, 3

❒ **MVS/XA address space:**

This diagram is not in proportion

The area above The Line is 127 times the area below The Line

The 20KB low System Area is 1/50th of 1 MB, or 1/800th of the area below The Line


Extended Private User Area
System Area above the line
The Line (16 MB)
System Area below the line
Private User Area
System Area - 20KB

❒ **The goal is to put very little code and data below the line and to have the vast majority of programs and data reside above the line**

The Road to z/OS, 4

- ❐ **Other variations of MVS came along, to support enhanced hardware instructions and features, but the essence of address spaces did not change**

- ❐ **The next step in the evolution was OS/390 (Operating System/390) which is really a packaging of components**

- ❐ **OS/390 contains**

  - ♦ **MVS code <u>plus</u> a number of program products as a single package**

  - ♦ **Intent was to update every six months, keeping all the products in synch, thus simplifying the process of installing and upgrading systems and products (1st release was 3/96)**

  - ♦ **Products included with OS/390 (among others):**

    - ✗ SMP/E (for maintenance uses)
    - ✗ TSO/E
    - ✗ ISPF
    - ✗ High Level Assembler
    - ✗ BookManager
    - ✗ DFSMSdfp
    - ✗ Language Environment (LE)
    - ✗ TCP/IP
    - ✗ DCE (Distributed Computing Environment support)
    - ✗ OpenEdition / POSIX support (UNIX under MVS!)

  - ♦ **In addition, other optional products are available to be shipped in an OS/390 order, for an extra charge**

The Road to z/OS, 5

- In 2001, IBM made available new hardware, the first of the zSeries machines, that supported 64-bit addresses
    - So now address spaces can be as large as 64-bit addresses allow
- A new operating system, z/OS, was announced to support the new hardware
- But z/OS is based on OS/390 - there is a solid continuity here
    - Old code can still run under z/OS, even code compiled and linked under earlier operating systems over 35 years earlier
    - To use new features, of course, you need to rewrite, recompile, and rebind
    - There are still address spaces, just larger and organized slightly differently
    - There is still an MVS component, a TSO component, and so on
- The last release of OS/390 was V2R10, available September 2000, the first release of z/OS was available March 2001
    - The announced intent is to slow the release schedule to once a year after V1R6 is available

The Road to z/OS, 6

❒ **Some of the issues around establishing a 64-bit address space are resolved this way**

- ♦ **The size of the low System Area is increased to 24KB**

- ♦ **The previous limit of 2 GB is now called <u>The Bar</u>**

  - ✗ So programs or data can reside

    - ➢ Below The Line

    - ➢ Above The Line but below The Bar

    - ➢ Above The Bar (data only, currently, no programs)

❒ **A 64-bit address space allows for a maximum address of 18,446,744,073,709,551,615**

- ♦ **That is, a 64-bit address space is 8,589,934,592 <u>times</u> the size of a 31-bit, 2 GB address space**

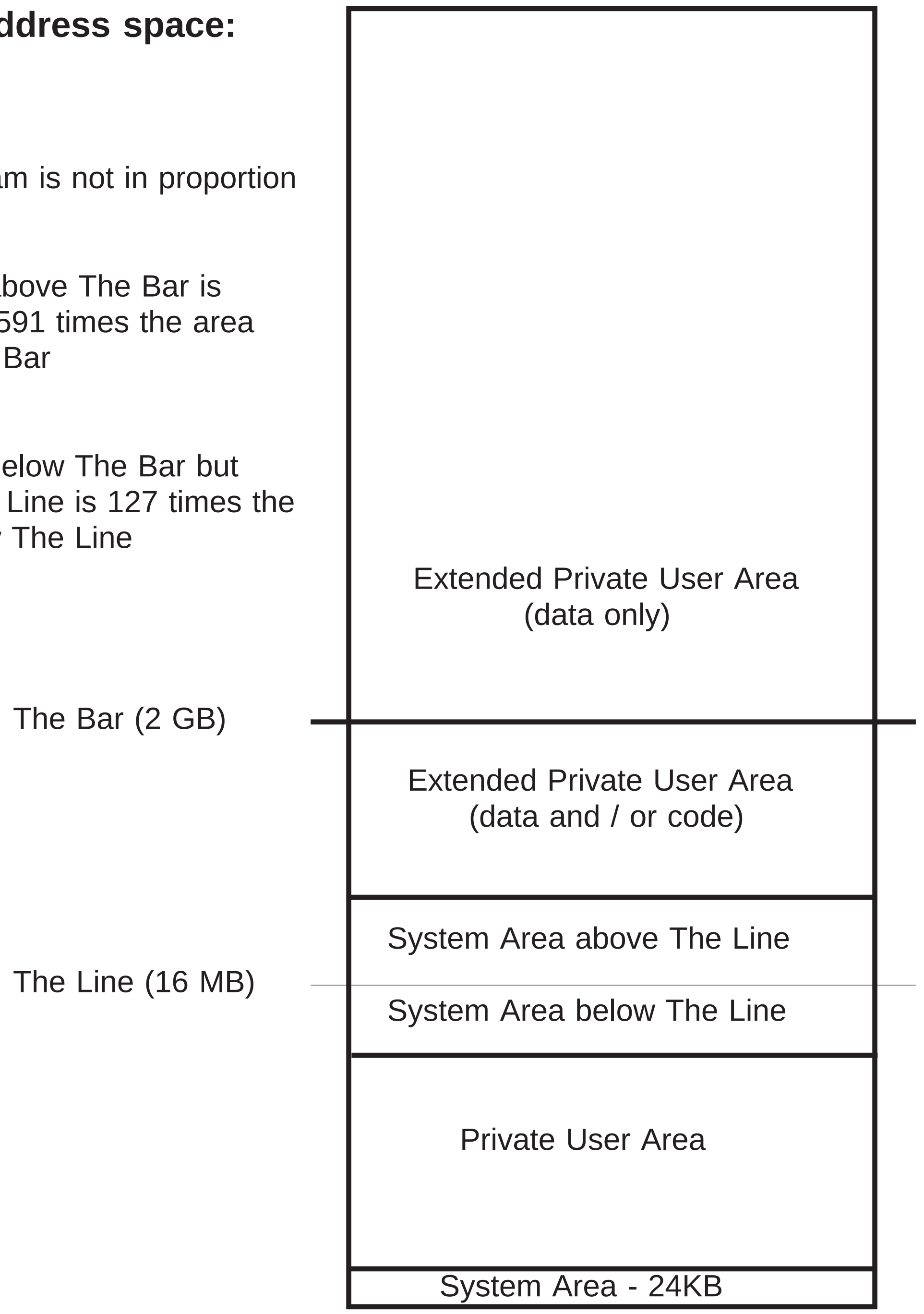
The Road to z/OS, 7

❒ **z/OS address space:**

This diagram is not in proportion

The area above The Bar is 8,589,934,591 times the area below The Bar

The area below The Bar but above The Line is 127 times the area below The Line


Extended Private User Area
(data only)
The Bar (2 GB)
Extended Private User Area
(data and / or code)
System Area above The Line
The Line (16 MB)
System Area below The Line
Private User Area
System Area - 24KB

❒ **Each job runs in its own address space, so now we move on to explore the management of jobs in z/OS ...**

Job Management

Job

✗ A unit of work to be run in the batch; one or more programs to be run in sequence

Job Queue

✗ An ordered collection of jobs

Job Class

✗ A one character code (A-Z, 0-9; 36 possibilities) assigned to each job

Job Priority

✗ A numeric value, 1-15 (1-13 for JES3 environment), that describes the relative importance of jobs within their job class (the higher the job priority number, the more important the job)

Job Management, 2

<u>Job Control Language (JCL)</u>

- ✗ A specification language used to describe jobs (work to be done) in terms of what resources are required, in what order, and under what conditions various work should get done

- ✗ JCL is written as a series of statements

<u>Job Stream</u>

- ✗ A collection of JCL and card [-image] input data read into the system for placement on the job queue

<u>SPOOL</u>

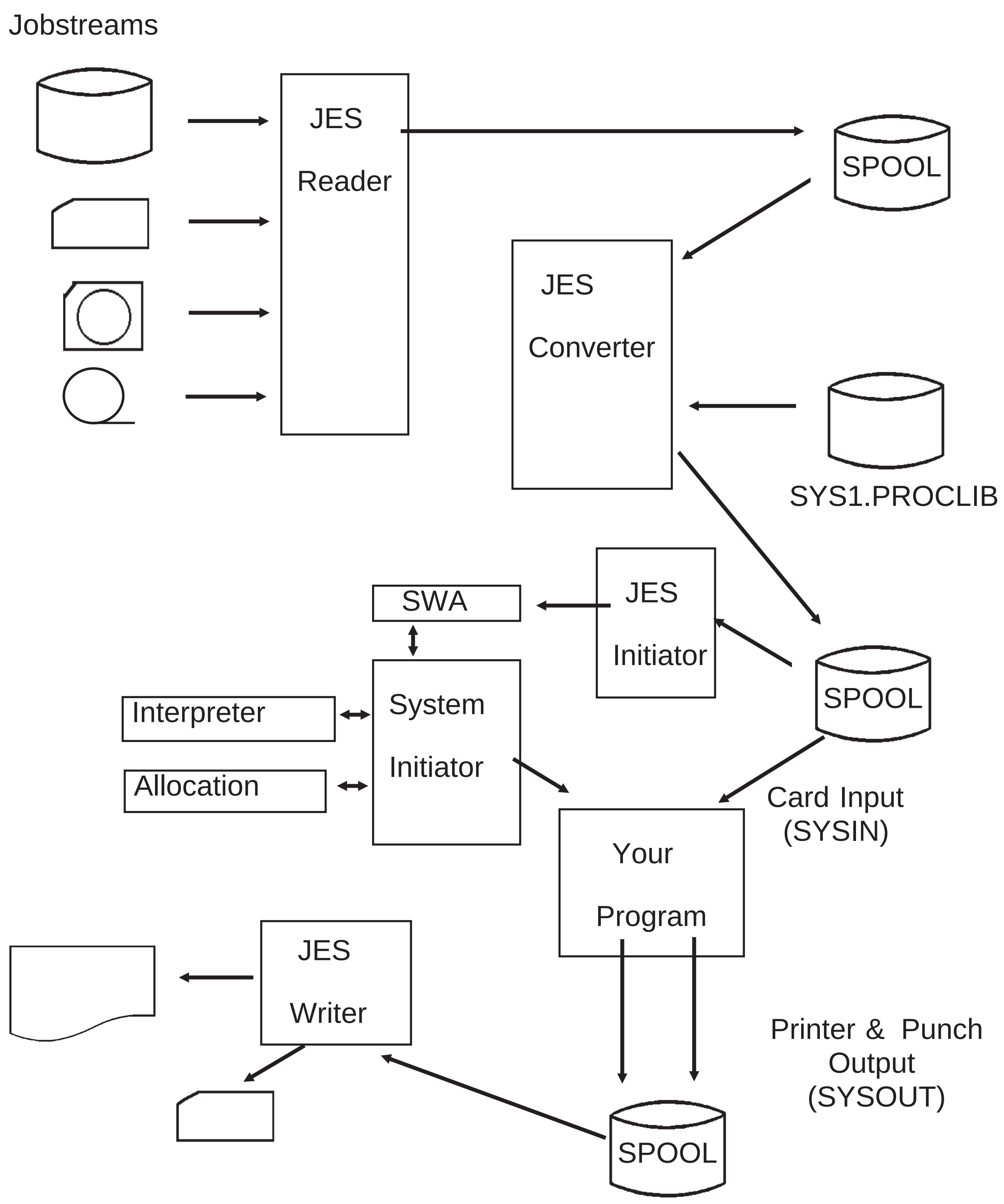
- ✗ Simultaneous Peripheral Operations On-Line

  - ➢ Using DASD work space to simulate the presence of multiple card readers, card punches, and printers

  - ➢ Thus allowing many jobs to be producing reports concurrently, even if you only have one printer

- ✗ The SPOOL area of DASD is also used to hold jobs in the queue waiting for work ...

z/OS Work Flow



Job Entry Subsystem

Reader

♦ **Reads job stream, puts on Job Queue by priority within class**

Converter

♦ **Converts free form JCL into control blocks on Job Queue**

Initiator

♦ **Selects which job to run next, based on class and priority**

Allocation

♦ **Creates the required environment for executing the job**

While the program runs, the SPOOL routines

♦ **Replace unit record I/O requests with I/O to SPOOL**

Deallocation

♦ **Frees resources on completion of step and job**

Writer

♦ **Transcribes SPOOLed output to printer or punch**

Job Purge Routine

- ❒ When the last line has been printed and the last card punched, the purge routine is invoked
  - ♦ Removes job JCL, and all SYSIN-type and SYSOUT-type records from SPOOL
  - ♦ Frees that SPOOL space to be used by subsequent jobs

- ❒ Note that there are two versions of JES: JES2 and JES3
  - ♦ The differences need not concern us here

How JCL Describes Resources

```
//jobname          JOB    (accountnginfo),prgrmrname,TIME=(min,sec),CLASS=x
//STEP1            EXEC       PGM=ISDED01,PARM='YNOFOUT/'
//STEPLIB           DD        DSN=DFIR.PROD.LOADLIB,DISP=SHR
//TRANSIN           DD        *
3560199227768
3568834990022                    sysin-type data (instream data)
4492445502367
•
•
•
//TRANSOUT          DD        UNIT=SYSDA,DISP=(,PASS),SPACE=(TRK,(25,10))
//MASTREF           DD        DSN=DFIR.CUST.MASTRFLE,DISP=SHR
//STEP2            EXEC       PGM=ISDUPDT,PARM='TEST/'
//GOODTRAN          DD        DSN=*.STEP1.TRANSOUT,DISP=(OLD,DELETE)
//MAST              DD        DSN=DFIR.CUST.MASTRFLE,DISP=OLD
//LOGTPE            DD        DSN=DFIR.APLILOG(+1),UNIT=TAPE,DISP=(,CATLG)
//LOGRPT            DD        SYSOUT=M
//UPDRPT            DD        SYSOUT=A,COPIES=2
//SYSUDUMP          DD        SYSOUT=D
```

The Allocation Process

Job Allocation

- **Step Allocation**
  - ✗ Units, volumes, data sets, DASD space; then program fetch loads in the program

$\boxed{\textbf{STEP EXECUTION}}$

- **Step Deallocation**
  - ✗ Data set disposition processing

Job Deallocation

- **Final data set dispositions**
- **Indicate SYSOUT-type data available for processing**

JCL Operations

❒ JCL describes the work to be done in a job through statements that are categorized into operations

❒ There are four major JCL operations, each described in separate JCL statements:

JOB statement

- ♦ Indicate the start of the JCL for a job; assign job class (which initiators can service this job), and some other basic descriptive information

EXEC statement

- ♦ Indicate step boundaries; each step runs one program

DD statement

- ♦ Data Definition; one for each data set resource the program in a step will need

OUTPUT statement

- ♦ Describe SYSOUT-type processing characteristics for some data sets

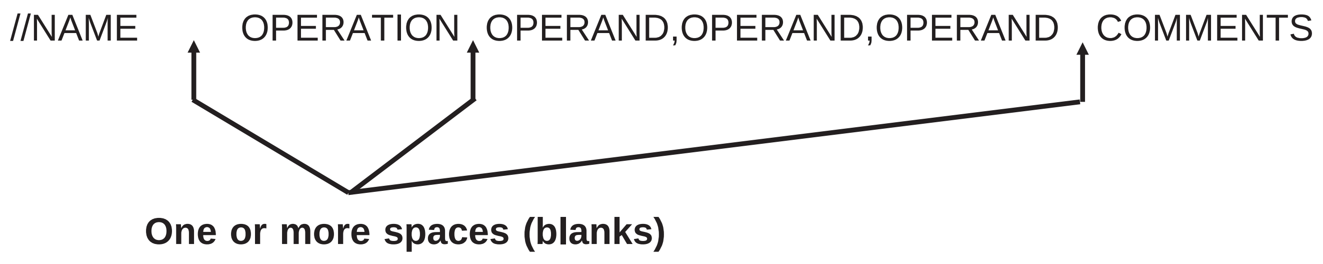
❒ Every job has one JOB statement followed by an EXEC statement for each program to run

❒ Each EXEC statement is followed by the DD statements that describe the data sets the program run in that step will use

❒ OUTPUT statements and their placement are described later

JCL Statement Format

Columns 1-71:



❐ **NAME: 1-8 characters from A-Z, 0-9, $, #, @; first not numeric**

Operation	**Name field**
JOB	**jobname**
EXEC	**stepname**
DD	**ddname**
OUTPUT	**outputname**

❐ **OPERANDS: Positional,Keyword**

JCL Statement Format, 2

❒ **Special formats:**

//*	**—**	**Comment statement**
//	**—**	**Null statement**
/*	**—**	**SYSIN data delimiter**

❒ **Note that there are other operations, some of which we will be discussing later in the class**

JCL Coding Rules

❒ **JCL is generally coded in uppercase**

- ♦ **Exception: when coding parameters to access files in the Hierarchical File System (HFS), which is not covered in this course**

❒ **Code up to column 72, then continue a statement, if necessary, on the next line as discussed on the next page...**

JCL Continuation

```
//NAME     OPERATION    OPERAND1,OPERAND2,
//     OPERAND3,OPERAND4,OPERAND5,
//     OPERAND6,    comments may go one or more spaces
//     OPERAND7,    after a comma
//     OPERAND8
```

❒ Continued statement must begin somewhere in columns 4-16, inclusive

JOB Statement Format

```
//jobname    JOB      (accounting info),progammer-name,
//             CLASS=x,MSGCLASS=y,
//             NOTIFY=userid,TIME=(min,sec),
//             TYPRUN={HOLD|SCAN}
```

- *accounting info* **- up to 143 characters, installation choice**
- *programmer name* **- up to 20 characters, installation choice**
- **CLASS is job class, implying which initiators may run this job**
  - ✗ Installation runs some number of initiator address spaces for running batch jobs; each initiator is assigned to handle particular sets of job classes
- **MSGCLASS is SYSOUT class, specifying where printed / punched output should go**
- **NOTIFY may specify a "***userid***" or "***node***.***userid***" ("node" option not supported in JES3)**
- **TIME special values: 1440 or NOLIMIT (both mean unlimited time), MAXIMUM (allows job to run up to 357,912 minutes - about 8 months)**
- **Omit TYPRUN normally; SCAN checks for JCL errors, HOLD keeps initiator from selecting job until explicitly released**

JOB Statement, continued

Examples

```
//MYWAY       JOB      (432,'RDD-343',NOXIOUS),JONES,
//     CLASS=A,MSGCLASS=H,NOTIFY=SJONES
.
.
.
```

```
//YOURWAY     JOB      (TR409,63),'O''NEIL',
//     NOTIFY=DEPT53.SSMITH,
//     TIME=2
.
.
.
```

❒ **Note: in JES3 systems, jobclass is specified on a JES3 MAIN statement, and job classes under JES3 can be up to 8 characters long**

Example:

```
//ANYWAY      JOB      (TRNG00P0),WIMP,
//     MSGCLASS=X,NOTIFY=WIMP
//*MAIN    CLASS=TSTHTEST
```

EXEC Statement Format

```
//stepname      EXEC      { PGM=programname
                          | PROC=procedurename
                          | procedurename}[,]
//            PARM=----------------,
//            TIME=(min,sec)
```

❒ **You must specify one of PGM=, PROC=, or just a name (which is then assumed to be a "*procedurename*")**

"programname"

♦ **System will look for this name in the directory of the library of executable programs called SYS1.LINKLIB (or its extensions)**

"procedurename"

♦ **System will look for this name in the directory of the library of pre-coded JCL called SYS1.PROCLIB (or its extensions)**

❒ **PARM is any string up to 100 characters long to be passed directly to the program being run**

❒ **TIME has the same possibilities as for the JOB statement, plus you may code TIME=0 which means use any time remaining from the previous step**

EXEC Statement, continued

Examples

```
//STEP1          EXEC     PGM=ISDR01R
```

```
//STEP2          EXEC     PGM=XYZARG,
//     PARM='DEPARTMENT 56, SAN JOSE'
```

```
//STEPX          EXEC     PGM=SORT,TIME=(12,30)
```

```
//LOUSY          EXEC     PGM=BIGRPT,PARM='FINAL RE
//             PORT ON STUDIES DONE IN JANUARY'
```

- **Note continuation of quoted string**
  - ✗ string coded up to (and including) column 71
  - ✗ continuation must have '//' in first two columns and continued text begins exactly in column 16

```
//CREDIT         EXEC     PROC=CPR43
```

```
//DISPUTE        EXEC     DSP567
```

Computer Exercise: OSWTO

Setup for all class labs:

Using ISPF option 6, enter the following command:

```
===> ex   '             .train.library(a700strt)'   exec
```

and press Enter.

This will cause the setup process to run. You will be prompted for a high level qualifier for your data sets. Unless the instructor tells you otherwise, use your TSO userid (the process is set up to use this as a default anyway). Press Enter.

The setup process will create a library for you to hold your JCL for the labs. The library name is <hlq>.TR.CNTL, where "<hlq>" is replaced by the high level qualifier you entered in response to the setup's prompt. This process also places a couple of members in your library you will need for various labs.

The lab ...

In your <hlq>.TR.CNTL data set, create a member called JCLEX01 to hold the Job Control statements necessary to run one job with two steps.

Reminder: to create a new member just use ISPF option 2 (edit); editing <userid>.TR.CNTL(JCLEX01) will create the member in your library, showing you an empty screen, ready to type.

First copy in member JOB at the front.

[From the command line enter: ===> *copy job*]

Next, since the program we are going to run is not stored in the standard system program library, we need to tell allocation where to find the program. After your JOB statement, and before any other JCL, code a statement like this:

```
//JOBLIB  DD  DISP=SHR,DSN=___________.TRAIN.LOADLIB
```

We'll discuss what this means later in the course.

Each of the two jobsteps should run the same program: OSWTO. So code two EXEC statements, each specifying PGM=OSWTO.

This program accepts from 1 to 25 characters (inclusive) from the PARM field on the EXEC statement and copies this data to the system message dataset (your JCL listing).

If you don't supply a value for the PARM field, the program will "blow up". On the other hand, if you supply more than 25 characters in the PARM field, the program will also "blow up".

So, on each step pass to the program, through the PARM field, your name (or userid) and the step name.

Do not run the job just yet - we have some more to talk about first.
(BUT... take a look at the next few pages for more info.)

This page intentionally left almost blank.

Clues for Writing JCL

❐ **Coding JCL is often a matter of "reading between the lines" of the information you are given, to translate this into JCL statements - looking for clues, as it were**

❐ **In the course of several exercises, we will sumarize pointers that are useful in most situations**

JOB Statements

♦ **Installation standards, and any JOB Statement generating edit macros normally provide you with all the information you need**

```
//JOBname   JOB   (acctng),pgmr_name,CLASS=x,MSGCLASS=y[,
//       REGION=nn{K|M}[,TYPRUN={SCAN|HOLD}][,TIME=(min,sec)]   ]
```

✗ JOBname - see installation standards reference

✗ Accounting info - installation specific

✗ Programmer name - up to 20 characters; installation specific or programmer choice

✗ CLASS= - installation specific job class

✗ MSGCLASS= - installation specific SYSOUT class for JCL listings

✗ TYPRUN=SCAN - does a basic JCL syntax check, does not actually run the job

✗ TYPRUN=HOLD - holds job until explicitly released; used to run job in off shift, or to hold until another job has run

✗ REGION= - job dependent, if required; virtual storage necessary to run the job

✗ TIME= - minutes and seconds to allow the JOB to run before cancelling; use for testing, not production

Clues for Writing JCL, 2

<u>EXEC Statements for Running Programs</u>

- ♦ **Need an EXEC statement for each program to run in a job; the order of the EXEC statements is the order the programs will be run in**

```
//stepname   EXEC   PGM=program_name[,PARM='   '][,TIME=][,
//      REGION=nn{K|M}]   ]
```

- ✗ Stepname - installation standard or programmer choice
- ✗ PGM= - name of program to run
- ✗ PARM= - up to 100 characters of parameter information; program specific; <u>only code if you are told the program expects or needs certain PARM or parameter data, and you are also told what data to code</u>
- ✗ TIME= - minutes / seconds this step is allowed to run; only code for testing, never production
- ✗ REGION= - program dependent, if required; virtual storage necessary to run the step

❐ **ALSO: you need to know where the programs are found**

- ♦ **If all programs are found in "the system libraries" or "the link list", then you do <u>not</u> need JOBLIB or STEPLIB statements**
- ♦ **If a program is found in a particular <u>library</u>, you need to code**

//STEPLIB   DD   DSN=<u>library_name</u>,DISP=SHR

<u>or</u>

//JOBLIB   DD   DSN=<u>library_name</u>,DISP=SHR

- ♦ **Place a STEPLIB after the EXEC statement it relates to; place a JOBLIB after the JOB statement**

Section Preview

❐ **Running Jobs**

- ♦ **The Work Load Manager**
- ♦ **The SCHENV parameter**
- ♦ **Submitting Jobs**
- ♦ **Monitoring jobs**
- ♦ **Looking at Job output**
- ♦ **Running a Job (Machine Exercise)**

The Work Load Manager

❒ **The Work Load Manager (WLM) is a relatively new component of the operating system that provides a sophisticated method of balancing work in a z/OS system**

❒ **To use WLM, your installation creates a number of profiles called "scheduling environments"**

- ♦ **Each scheduling environment describes resources for specific types of jobs**
  - ✗ Typically, specific DB2 subsystems, but the concept is very general
- ♦ **To place a job into a particular WLM scheduling environment, code a SCHENV parameter that names the desired scheduling environment on the JOB statement**

❒ **The WLM runs in an address space that monitors system performance and how jobs are doing in terms of meeting the goals of their environments**

- ♦ **The WLM can start and stop address spaces for running the various jobs it is responsible for, balancing the mix of work going on**
- ♦ **Each job will be scheduled to run on a system that has the required resources and settings**

The SCHENV Parameter

- ❒ **Code a SCHENV parameter on a JOB statement to identify the scheduling environment the job should run in**
  - ♦ **Scheduling environment names are 1-16 characters long**
    - ✗ Alpha, numeric, national (@,#,$), or underscore (_)
      - ➢ An underscore character may not be the first nor last character in the name, and if an underscore is present, the name must be bounded by single quotes
- ❒ **Scheduling environment names are installation specific**
  - ♦ **There are no default or supplied names provided with your system**

Example

```
//JDE3301  JOB  (AA-22),PRD12,SCHENV=OEPLEX
```

SUBMIT — EDIT / BROWSE / VIEW Primary Command

❐ **If you are browsing, viewing, or editing JCL under ISPF/PDF, use the SUBMIT command to run a job in the batch**

```
===> SUBMIT          or just          ===>SUB
```

- **What gets submitted is the data you're currently working with**

- **If the name on your JOB statement is just your userid, SUBMIT will prompt you with:**

  ENTER JOBNAME CHARACTER(S)

- **You may enter <u>one or two characters</u> that will be appended to your userid on the JOB statement; this helps you to assign unique jobnames to each job you submit**

- **Then you will see the message**

  JOB jobname (JOBnnnnn) SUBMITTED
  ****

- **Jot down the JOBID (that is, the job number: nnnnn) (you can omit leading zeros)**

- **When you see the three asterisks, press <Enter>; your job is now on the batch queue waiting for an initiator**

NOTIFY

❒ If your JOB statement specified the NOTIFY parameter, you can do other work while your job runs

❒ When the job is complete, the system will notify you:

- When you press <Enter> (or any function key) for your work in progress, your screen will be blanked out (relax: you haven't lost any data) and the job completion message will be displayed followed by three asterisks (***)

- Press <Enter> and the screen will be restored to its pre-notify content

Examining Job Output

- ❐ **When a job is complete and you want to see how it did, you have a variety of tools to choose from: ISPF 3.8, SDSF, FLASHER, IOF, (E)JES**
  - ♦ **ISPF 3.8 (Outlist) is available on all z/OS and OS/390 systems; the other products must be purchased separately**
    - ✗ However, in recent years we have found a few shops that remove or change the ISPF 3.8 option; in this case you may need to use the techniques discussed later in this course
- ❐ **All these options let you examine jobs on the queues**
  - ♦ **And to dispose of the various listing outputs by keeping, printing, or deleting**
- ❐ **We start with ISPF option 3.8 ...**

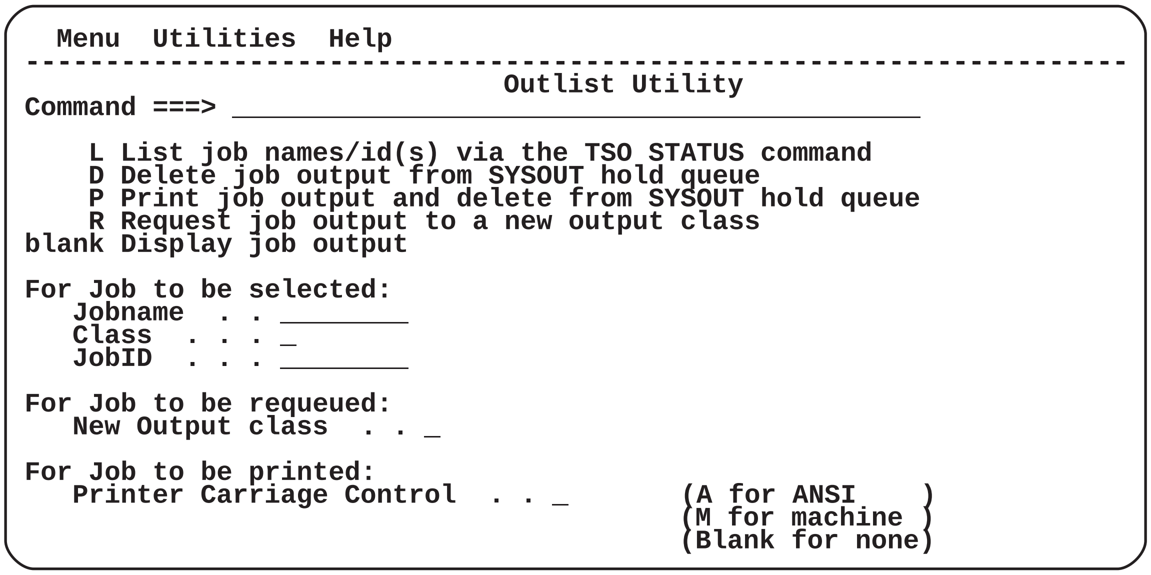
Monitoring Jobs

A. - Use the ISPF Command shell (Option 6)

- **Issue TSO STATUS command:**

```
ST

ST jobname

ST jobname(Jnnn)
```

B. - Use Option 3.8 (Outlist)



```
  Menu  Utilities  Help
---------------------------------------------------------------------
                              Outlist Utility
Command ===> ___________________________________________

    L List job names/id(s) via the TSO STATUS command
    D Delete job output from SYSOUT hold queue
    P Print job output and delete from SYSOUT hold queue
    R Request job output to a new output class
blank Display job output

For Job to be selected:
   Jobname  . . ________
   Class  . . . _
   JobID  . . . ________

For Job to be requeued:
   New Output class  . . _

For Job to be printed:
   Printer Carriage Control  . . _       (A for ANSI    )
                                         (M for machine )
                                         (Blank for none)
```

Monitoring Jobs, continued

B. - Using Option 3.8 (Outlist), continued

- ♦ **Type in 'L' on the Command line and get back one of:**

  - ✗ **List of jobs (JOBNAME's & JOBID's) with status:**

    WAITING FOR EXECUTION

    EXECUTING

    ON OUTPUT QUEUE

  - ✗ **Or one of these messages:**

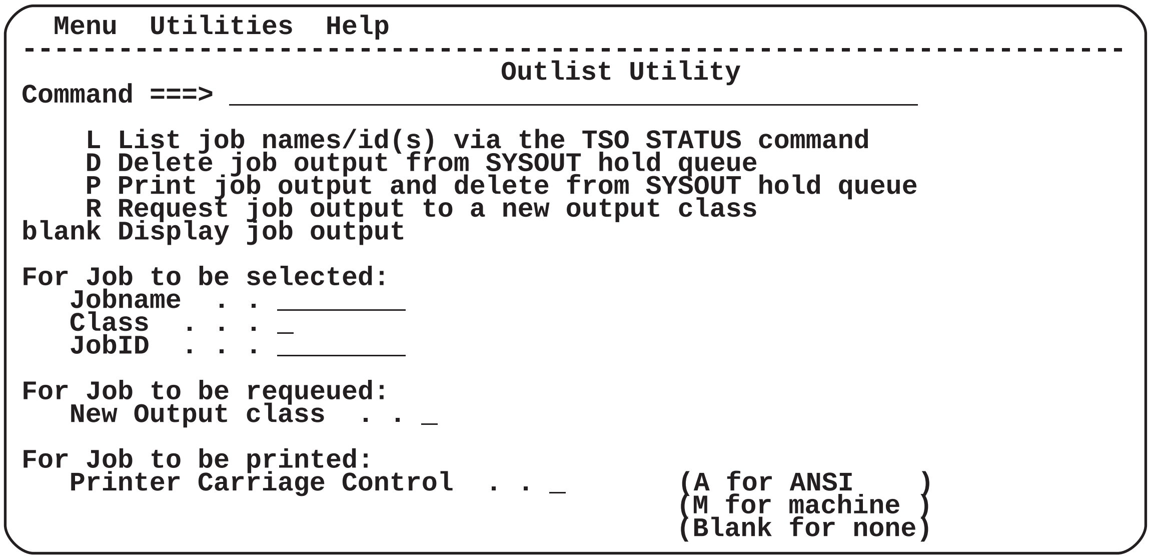
    JOB NOT FOUND

    NO JOBS FOUND

❐ **The same messages are retrieved from the TSO STATUS command and the ISPF 3.8 panel with the L command**

Working With Job Output

Under Option 3.8 (OUTLIST)



```
   Menu  Utilities  Help
 -------------------------------------------------------------------
                               Outlist Utility
Command ===> ___________________________________________

    L List job names/id(s) via the TSO STATUS command
    D Delete job output from SYSOUT hold queue
    P Print job output and delete from SYSOUT hold queue
    R Request job output to a new output class
blank Display job output

For Job to be selected:
   Jobname  . . ________
   Class  . . . _
   JobID  . . . ________

For Job to be requeued:
   New Output class  . . _

For Job to be printed:
   Printer Carriage Control  . . _       (A for ANSI    )
                                         (M for machine )
                                         (Blank for none)
```

- **Fill in Jobname**

  ✗ If Jobname not unique, also type in JobID as

  J*nnn*

- **Then, in the command line, type in one of:**

ƀ	**to display output on the screen**
D	**to delete the job output**
P	**to print the job output (submits a job to print it out)**
R	**to requeue output: place on different output class for printing or other disposition** **— must then specify a new output class on the 3.8 panel**

Working With Job Output, continued

Still using ISPF 3.8

- **Under the display option, you can now process your output as if you were under Browse or View: FIND, scrolling commands, and so on, all work**

- **When done looking at the output, F3 ('END' command) returns you to the 3.8 panel, and you can then decide to leave the output on the queue, to delete it, or to requeue it**

Computer Exercise: Run a Job

Edit your job to run OSWTO. If necessary or appropriate, code a SCHENV parameter on the job statement.

Submit the job. Use ISPF 3.8 to examine the output.

This page intentionally left almost blank.

Section Preview

❐ **Introduction To Data Management**

- ♦ **Data Management Terms**
- ♦ **File References**
- ♦ **DCBs / ACBs**
- ♦ **DDNAMEs**
- ♦ **DCB Parameters**
- ♦ **SYSIN-type data and SYSOUT-type data**
- ♦ **A JOB with DD statements (Machine Exercise)**

Data Management Terms

Field

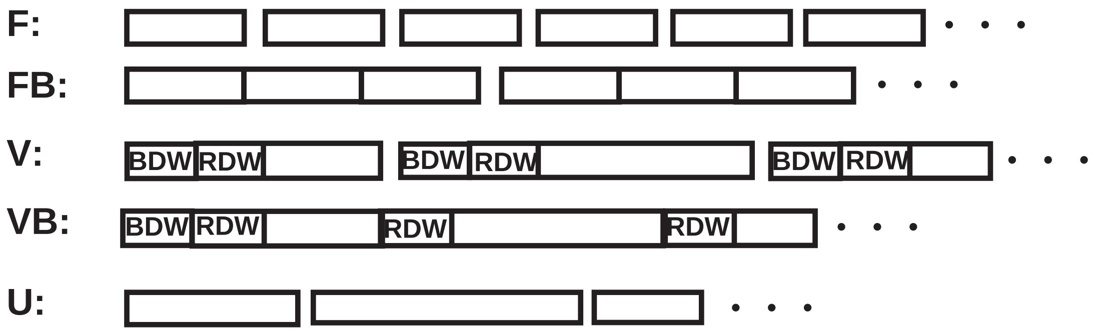
- **The smallest unit of data organization (1 or more bytes in length)**
  - ✗ Examples: Customer Name, Part Number, Hourly pay rate, Date of invoice, Mortgage type, Status flags

Logical Record

- **A collection of related fields; a logical record represents some object or state of being**
  - ✗ Examples: Customer Record, Personnel Record, Transaction Log Record

Supported Logical Record Formats:

Fixed Length, Unblocked or Blocked (F, FB)
Variable Length, Unblocked or Blocked (V, VB)
Undefined (U)



BDW: Block Descriptor Word **RDW: Record Descriptor Word**

Data Management Terms, 2

Physical Block

- **1 or more logical records: the unit of recording on media, also the unit of data transmission between memory and the media**

Data Set = File

- **A collection of related records**
  - ✗ Examples: Customer master file, Transaction input file, Inventory report
  - ✗ Also spelled with no intervening space: dataset

Volume

- **Physical unit of magnetic or optical media**
  - ✗ Examples: disk pack, tape reel, tape cartridge, laser disk (video disk)

Unit

- **Machine that holds a volume for reading and / or writing**
  - ✗ Examples: disk drive, tape drive

Data Management Terms, 3

Data Set Organization

The method in which data records are organized (written) on media

- **Sequential**

  Physical sequence, not necessarily logical sequence
  To access the 10,000th record must read all preceding 9,999
  Supported on all media (paper, cards, tape, ...)
  Literature references as SAM, BSAM, or QSAM
  (also see VSAM ESDS, on next page)

- **Partitioned**

  'Libraries'
  Directory at front, containing names of members and address of members on disk
  Members: small sets of records, such as source programs, control statements, or data

- **Indexed Sequential** - **replaced by VSAM KSDS**
- **Direct** - **supplanted by VSAM RRDS**
- **Extended Sequential** - **supports RAID (Redundant Array of Independent Disks) technology**
- **HFS - Hierarchical File System** - **used to support z/OS UNIX**
- **PDSE - Partitioned Data Set, Extended** - **discussed later**

❒ **The above are collectively called non-VSAM data set organizations**

Data Management Terms, 4

Data Set Organization, continued

♦ **VSAM - Virtual Storage Access Method**

Collection of related organizations
Sharing some common physical recording techniques

➢ ESDS - Entry Sequenced Data Set

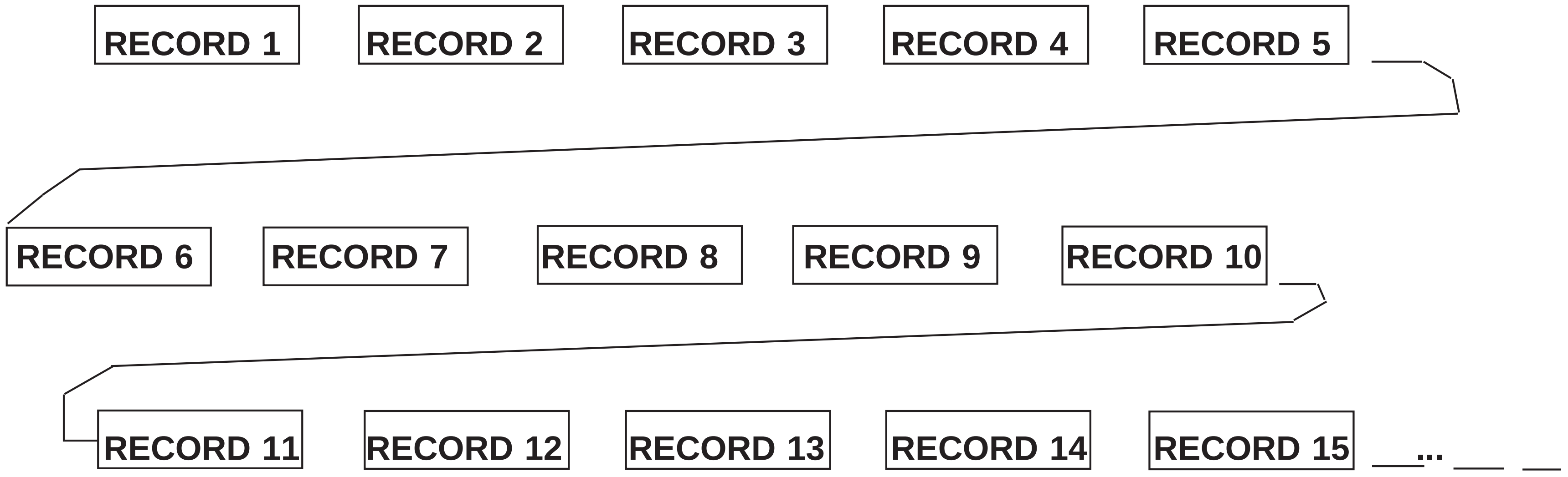
➢ KSDS - Key Sequenced Data Set

➢ RRDS - Relative Record Data Set

➢ LSDS - Linear Space Data Set

✗ Details beyond the scope of this course

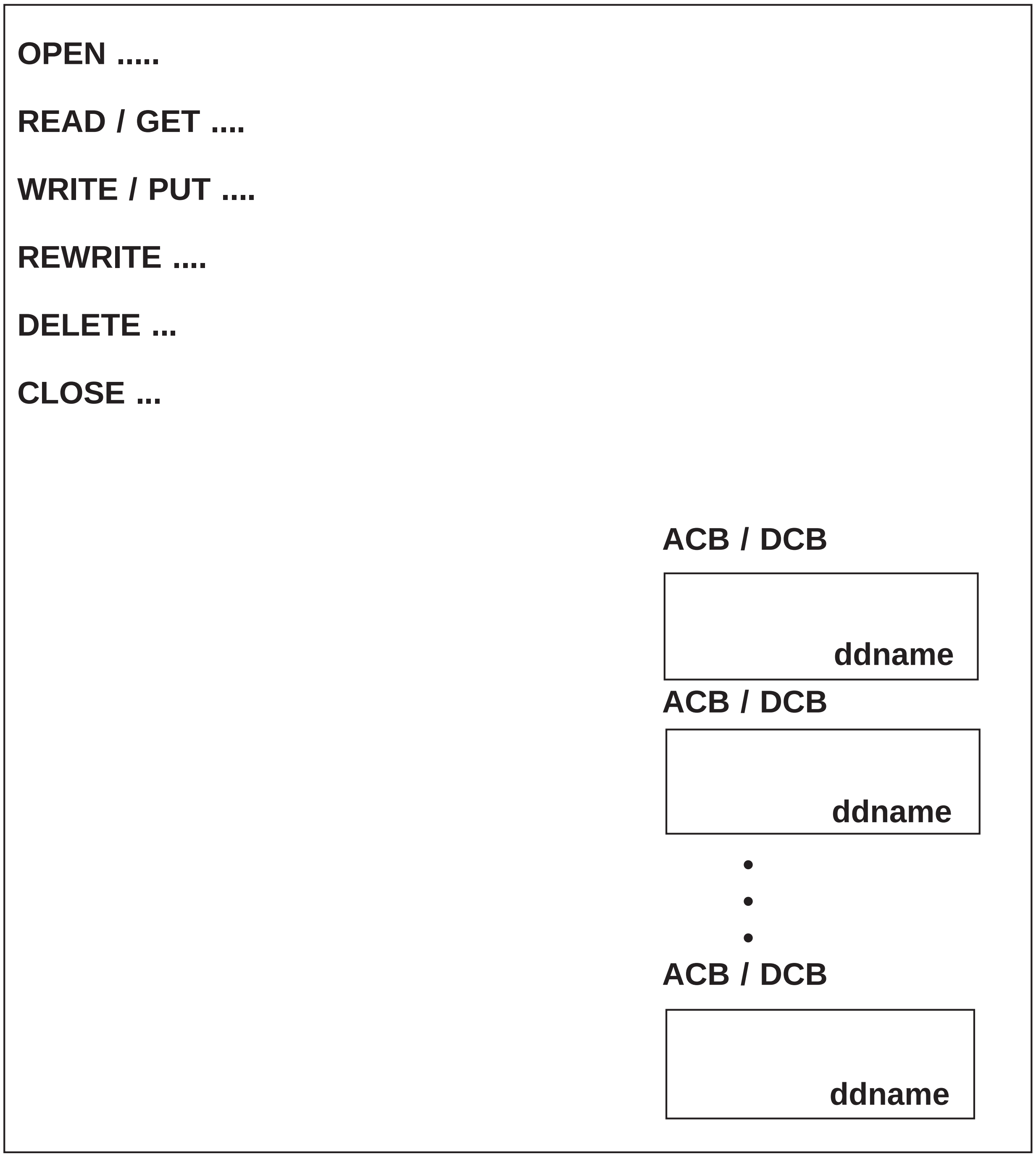
A Sequential Data Set



- Records stored in physical sequence
- Must be retrieved in physical sequence
- May be on any medium (Disk, Tape, Paper, Cards...)

File References

❐ Whenever a program references a file, the reference is really to a small control block that describes and represents the file: an ACB (for VSAM files) or DCB (for non-VSAM files)



DDNAMES

❒ **Although ACBs and DCBs have different formats, they perform the same functions: to describe the data set characteristics, and to provide a connection between an internal file name and an external data set through another name, a DDNAME:**

ACB / DCB

```
data set organization
access technique used
processing mode
record format
record size              block size
end of data routine address
error / exception routine addresses
work areas, status indicators
ddname: XXXXXXXX
```

•
•
•

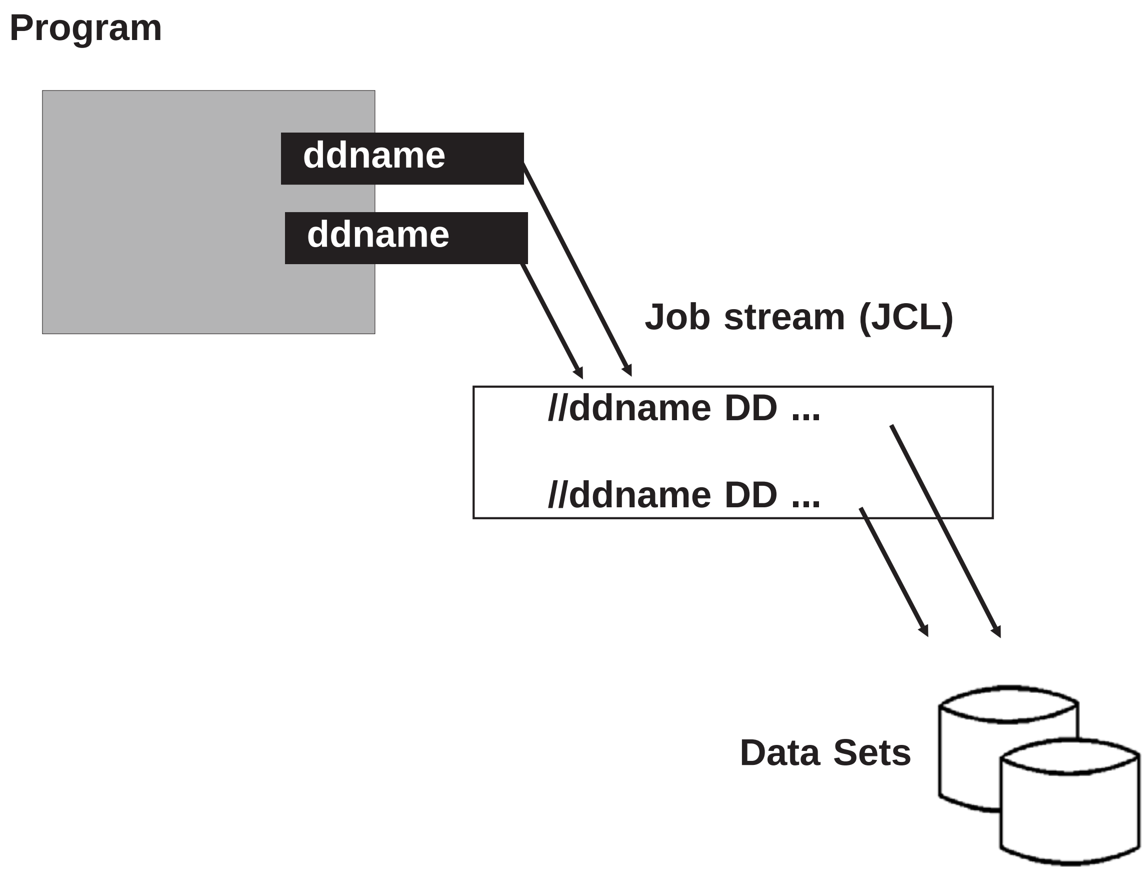
ACB / DCB

```
data set organization
access technique used
processing mode
record format
record size              block size
end of data routine address
error / exception routine addresses
work areas, status indicators
ddname: XXXXXXXX
```

DDNAMES and Data Set Names

- ❒ A <u>DDNAME</u> is a name (1-8 alphanumerics, first of which is not numeric) that can be mapped to a real data set

    - ♦ Using DD statements in JCL for batch jobs



- ❒ Then, program requests issued using the DDnames are applied against the data set(s) mapped to those respective DDnames

DCB Parameters

- **The various data characteristics stored in a DCB representing a file in a program are also stored in the label of the actual tape or disk data set, once the file is created**
  - **If the data set has standard labels**

- **These characteristics may also be specified in JCL, on the DD statement that relates to a file ...**

DCB Parameters, continued

❒ There is a large number of these parameters, but there are only a handful you might need to code:

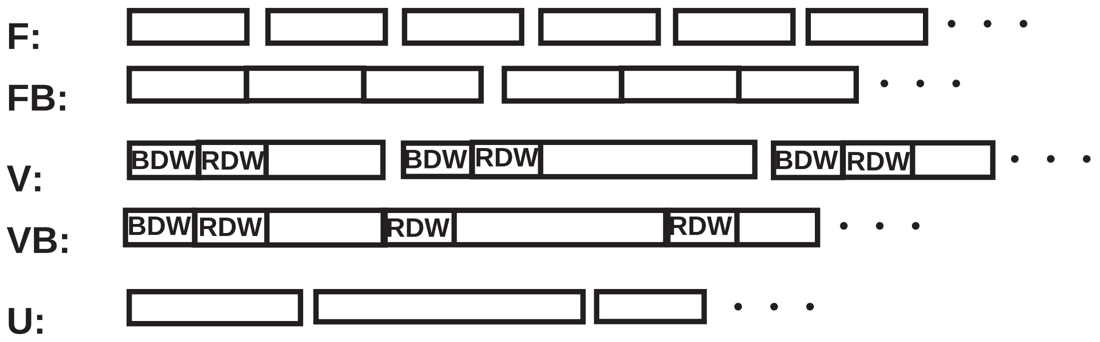
- ♦ LRECL - logical record length; the size of a record in bytes

- ♦ BLKSIZE - block size; the size of a physical block in bytes

- ♦ RECFM - record format; a code describing the format of records in a file; described earlier:

F	Fixed length, unblocked
FB	Fixed length, blocked
V	Variable, unblocked
VB	Variable, blocked
U	Undefined

Appending an 'A' indicates the data contains ANSI printer carriage control characters in the first byte



- ♦ DSORG - data set organization; usually PS (for physical sequential) or PO (for partitioned organization)

- ♦ TRTCH - for tape; COMP (hardware should compact the data) or NOCOMP (hardware should not compact the data)

Coding DCB Parameters in JCL

❒ **In most cases, you do not need to code DCB parameters in JCL**

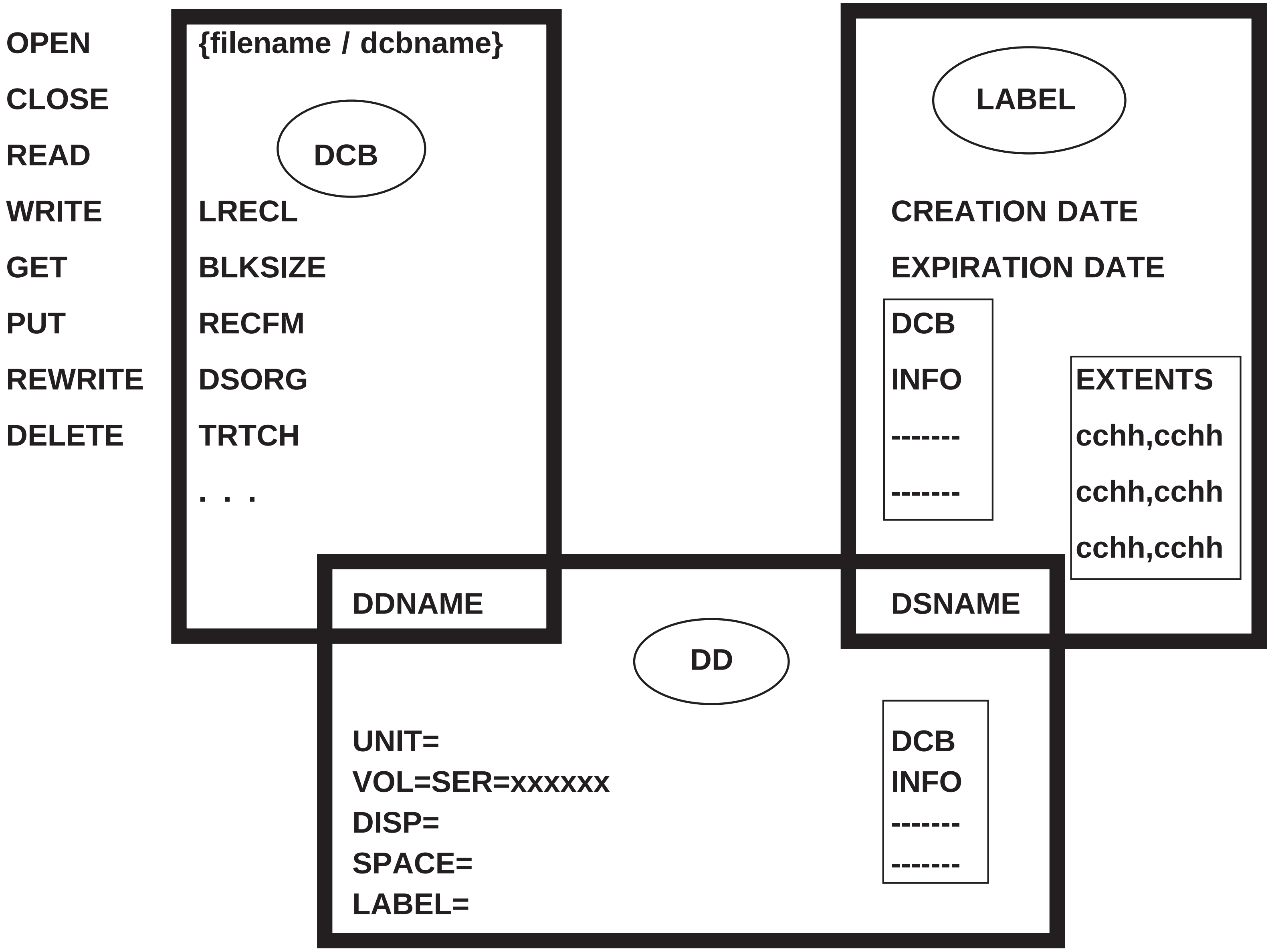
- ♦ **The values can be obtained by OPEN from the DCB in the program, on the file label, or by system defaults**

- ♦ **But, when you need to code them, they are just like any other keyword parameter, for example:**

```
//          LRECL=1200,RECFM=VB
```

❒ **We will explore when you need to code these parameters as we learn more**

Connections



Guidelines for Coding DCB Information in JCL

❐ **Nowadays, it is rare to code any DCB parameters in JCL, but if a program is written to be very general, the program may have omitted many DCB details in anticipation of filling in the blanks at run time via the JCL**

- ♦ **Using SMS can simplify DCB parameter coding even more (discussed later)**

❐ **Note that when a standard-labeled output or update file is CLOSEd by the program, the completed DCB is written out to the label for later use**

- ♦ **Subsequent programs reading the file can get all the DCB information from the label: no need to code DCB at all**

❐ **For SYSIN-type data (card image data), no need to code DCB parameters**

❐ **For SYSOUT-type data (printer / punch data) may need to code RECFM, LRECL, or BLKSIZE**

Guidelines for Coding DCB Information in JCL, continued

❒ **For input tape or disk files, let OPEN get DCB data from label of the data set**

Exceptions - may need to code DCB parameters for

- **Unlabeled tapes**
- **Non-standard labeled tapes**

❒ **For output tape files, code any missing DCB information on the DD statement**

- **Possibly LRECL, BLKSIZE, RECFM, TRTCH**

❒ **For output disk files, code any missing DCB information on the DD statement**

- **Possibly LRECL, BLKSIZE, RECFM, DSORG**

DD Statements - SYSIN-type Data

```
//TIMECRDS   DD   *
00938AA0101010400-RDS33
00939AA0101010425-RRD33
00942AB0101010450-RDS34
00945AB0101010380-NCO34
.
.
.
```

data in the input stream; in this case, looks like time card data that has been keyed in

```
//SYSUT1   DD   DATA
//COPYNPRT   PROC
//COPY       EXEC  PGM=IEBGENER
.
.
.
/*
```

data in the input stream; in this case, JCL that is being copied to a PROCLIB

```
//TRANSIN   DD   DATA,DLM='++'
Part235  received=50
/*334    shipped=100
A-2890A  shipped=99
//871    received=75
.
.
.
++
```

data in the input stream; in this case, inventory data where part numbers sometimes include "//" or "/*" (but never "++") in the first two positions

- ❒ Note that * and DATA are positional parameters
- ❒ SYSIN-type data is an input file that comes from the job stream rather than from a tape or disk

DD Statements - SYSOUT-type Data (Printer and Punch Output)

```
//SYSPRINT   DD   SYSOUT=A

//LISTING    DD   SYSOUT=E,COPIES=12          - multiple copies

//REPRT      DD   SYSOUT=M,OUTLIM=1000        - stop if exceed limit

//TRANSLOG   DD   SYSOUT=*                    - use MSGCLASS

//CUSTLIST   DD   SYSOUT=*,LRECL=133,RECFM=FA
                                              - if missing in program

//MISCLIST   DD   SYSOUT=Y,HOLD=YES           - operator must release

//PARTS      DD   SYSOUT=T,FREE=CLOSE         - spin off at close

//BIGLIST    DD   SYSOUT=C,SPIN=UNALLOC       - spin off at step end

//ENORMOUS   DD   SYSOUT=F,SEGMENT=250        - spin off segments

//DUEDATES   DD   SYSOUT=L,DEST=RMT23         - printer name
```

The general syntax is: SYSOUT=(c,writer,formsid)

```
//SPECIALS   DD   SYSOUT=(R,,4PLY)
```

❒ **When a file is described using the words <u>REPORT</u>, <u>LISTING</u>, <u>PRINTOUT</u>, <u>DUMP</u>, and similar terms, this usually means "SYSOUT=" is used to describe the file in JCL**

Reserved DDnames

❒ **Some DDnames have special meaning to the system:**

<u>SYSUDUMP</u>

- **Place to write formatted dump of user area, if program ABENDs ("blows up")**

```
//SYSUDUMP    DD     SYSOUT=D
```

<u>SYSABEND</u>

- **Same as SYSUDUMP + dump of nucleus!**

```
//SYSABEND    DD     SYSOUT=*
```

<u>SYSMDUMP</u>

- **Place to write unformatted, machine readable dump**

```
//SYSMDUMP    DD     DISP=(,CATLG),UNIT=SYSDA,
//    DSN=GTRDUMP.SYSMDUMP,SPACE=(CYL,(1,1))
```

<u>CEEDUMP</u>

- **Place to write formatted Language Environment (LE) dump**

```
//CEEDUMP     DD     SYSOUT=*
```

Reserved DDnames, 2

❒ There are also DDnames that are frequently used by the various language compilers for use at runtime; especially note

SYSOUT

- ♦ COBOL uses for output of DISPLAY statements and some debugging information

SYSIN

- ♦ COBOL uses for ACCEPT statement, PL/I uses for stream input files

SYSPRINT

- ♦ PL/I uses for stream output / print files

Reserved DDnames, 3

<u>**JOBLIB**</u>

- ♦ **Load module library to look for programs run in this job**

```
//JOBLIB      DD       DISP=SHR,DSN=...
```

  - ✗ **Must appear before any steps (EXEC statements) in the job**

<u>**STEPLIB**</u>

- ♦ **Load module library to look for programs run in this step**

```
//STEPLIB     DD       DISP=SHR,DSN=...
```

  - ✗ **Must appear in step for which it applies (*i.e.*, after EXEC statement, before next EXEC statement)**

❒ **When the system is looking for a program...**

- ♦ **it first looks in any STEPLIB libraries defined in the step (if there is no STEPLIB statement, it looks in the JOBLIB libraries, if any)**
  - ✗ JOBLIB is ignored for any step that contains a STEPLIB
- ♦ **if the program still can't be found then it looks in the system link libraries**
- ♦ **if the program still can't be found, the job fails**

Reserved DDnames, 4

- There are other reserved DDnames, with additional special functions; see the JCL reference manual; generally ...

  - Avoid creating application DDnames that begin with SYS, JOB, STEP

  - In a JES3 environment, also avoid DDnames that begin with JC, JES, and the names JOURNAL and JST

  - In addition to CEEDUMP, some other Language Environment DDnames may start with CEE

Computer Exercise: SYSIN And SYSOUT Files

Create the JCL Statements necessary to run a two step job. First, create new member JCLEX02 and copy in JOB statement as before; then code JCL statements as follows:

Step One

First code an EXEC statement to request that the program named LIF2F be run.

This program is found in the library __________.TRAIN.LOADLIB, so you will need to code a STEPLIB DD statement for this dataset (DISP=SHR, please).

LIF2F reads records from an input file, (using a DD statement named 'INDD') and copies them to an output file unchanged (using a DD statement called 'OUTDD'). So you will need DD statements with these names:

Code the INDD DD statement to pass three lines of card-image test data as input, such as:

```
THIS IS LINE ONE OF THREE
THIS IS LINE TWO OF THREE
THIS IS LINE THREE OF THREE
```

Code the OUTDD DD statement to send the output to a print (SYSOUT) file. The DD statement for this file must include the following parameters, since they are missing from the program:

```
LRECL=80,RECFM=F
```

Step Two

Run a program called IEBGENER (that is, code an EXEC statement for this program). IEBGENER needs DD statements with these DD names:

SYSUT1	— points to input file; in this case, point to a few card-image lines, such as:

```
'TWAS GORBY WHO THE BUREAUCRATS
DESPISED AND FEARED FOR MAJOR CHANGE.
SURPRISED THEY WERE TO FIND THE FACTS:
WITH OR WITHOUT HIM, CHANGE CAME.
```

SYSUT2	— points to output file; route this to a print file
SYSPRINT	— contains error or processing messages from running the program; route this to a print file
SYSIN	— points to a file that contains instructions for special processing, if any

Note that if no special processing is required, provide an empty file, *e.g.*:

```
//SYSIN    DD   *
/*
```

For this exercise, we will use this format for SYSIN

IEBGENER is found in the system load module libraries, so you do not need a JOBLIB or STEPLIB DD statement.

Note that a step can have any number of DD statements using the SYSOUT parameter (for example, SYSUT2 and SYSPRINT above) and the system will keep track of each one seperately.

Run this job.

(NOTE: see more clues on the following pages.)

This page intentionally left almost blank.

Clues for Writing JCL, continued

DD Statements

- **Code one DD statement for each file referenced by a program; the DD statements for a step follow the EXEC statement for that step**

//xxxxxxxx   DD   parameters

- **"xxxxxxxx" is a "ddname" and must be given to you, as follows**
  - ✗ "A DD statement named xxxxxxxx"
  - ✗ "A DD statement called xxxxxxxx"
  - ✗ "Using a DD statement of xxxxxxxx"
  - ✗ A reserved or implied DD name:
    - ➢ SYSUDUMP and / or CEEDUMP - for dumps
    - ➢ JOBLIB or STEPLIB - see bottom of page 382
    - ➢ SYSOUT - for COBOL DISPLAY and for certain utilities
    - ➢ SYSPRINT - for PL/I stream output; for all utilities, including IDCAMS and SORT
    - ➢ SYSIN - for COBOL ACCEPT; PL/I stream input; IDCAMS comand input; SORT control statement input; compilers source code to be compiled

- **"parameters" - follow the chain of clues and questions on the following pages ...**

Clues for Writing JCL, continued

DD Statements

//ddname   DD   parameters

- ♦ **“parameters” - for each DD statement, check ...**
  - ✗ If a file is described as being “in the input stream”, simply code an asterisk (*); this is SYSIN type data; done with this DD statement
    - ➢ NOTE: a file with a DDname of SYSIN may be SYSIN type data or it may be a disk or tape file; if it is a disk or tape file, you need to know the name of the file; in the absence of such information, assume the file is in the input stream
    - ➢ You do not need to code a /* statement unless your input data contains lines with “//” in columns 1 and 2; in this case, code “DATA” instead of an asterisk
  - ✗ If a file is described using any word like “listing”, “list”, “printout”, “report”, “dump”, “print file”, “SYSOUT”, “messages file”, code: SYSOUT=x; done with this DD statement
    - ➢ 'x' is the output class the report should be sent to; code '*' for the class to use the class from the MSGCLASS parameter on the JOB statement
    - ➢ May also code some related parameters for special cases: DEST= for a remote destination; COPIES= for multiple original copies; HOLD=YES to keep a report from printing until it is released; FREE=CLOSE to allow a report to begin printing before end of job
    - ➢ Occasionally may need to code the DCB parameters LRECL= or RECFM=; only code if explicitly told to do so

Section Preview

❒ **Tape and Disk Data Sets**

- ♦ **SYSIN and SYSOUT Data Sets**
- ♦ **Tape and Disk Data Sets**
- ♦ **Tape Layout**
- ♦ **DASD Concepts**
- ♦ **Data Set Naming Rules**
- ♦ **Units**
- ♦ **Volumes**
- ♦ **Catalogs**

SYSIN and SYSOUT Data Sets

❐ **SYSIN-type data sets and SYSOUT-type data sets have these characteristics:**

- ♦ **Not usually named**
  - ✗ Actually, the system assigns names for internal use, but you cannot reference these names
- ♦ **No labels**
- ♦ **Allocation uses SPOOL space to hold the data**
- ♦ **Only accessible by one program; when program ends, no other program can access (transient)**
- ♦ **OPEN may need some DCB parameters**

Tape and Disk Data Sets

❒ **All other data sets in z/OS are tape or disk type data sets, and these data sets have the following characteristics:**

- **Always named, either permanent or temporary type names**

- **Disk data sets have labels; tape data sets usually have labels, but not always**

- **Allocation locates existing data sets by identifying the type of media (UNIT) and which particular volume (VOLUME) they reside on**

- **Allocation selects a location for a new data set by being informed what UNIT and VOLUME to use**

    - ✗ New disk data sets must have a SPACE request

- **Allocation may access disk data sets on an exclusive or shared basis; (tape is always exclusive)**

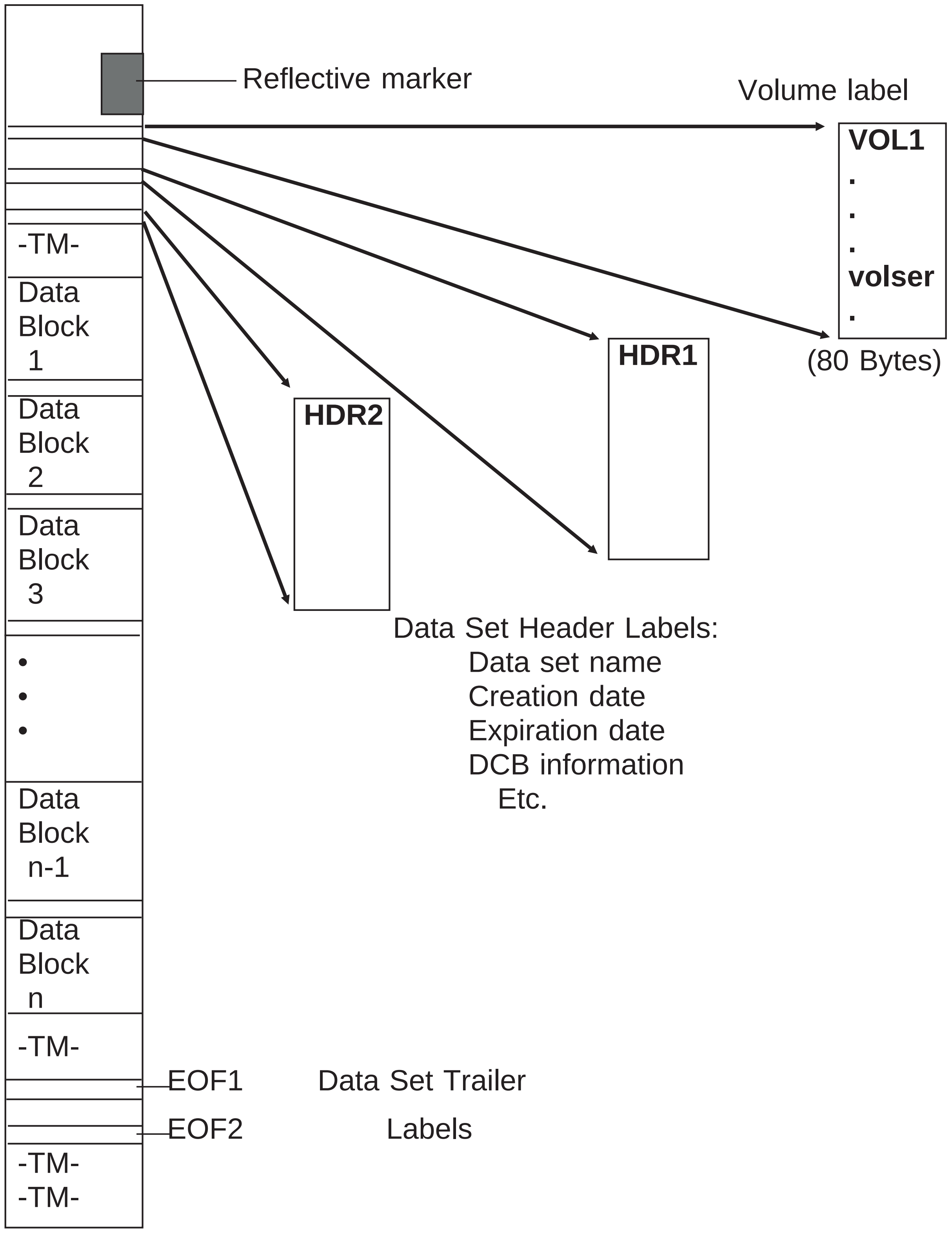
- **Data sets may persist across steps and jobs (or not)**

- **Data sets may be cataloged**

- **Data sets may be date protected**

- **OPEN may require some DCB parameters**

Tape Layout



Reflective marker
Volume label
VOL1
.
.
.
volser
.
(80 Bytes)
HDR1
HDR2
Data Set Header Labels:
Data set name
Creation date
Expiration date
DCB information
Etc.
-TM-
Data Block 1
Data Block 2
Data Block 3
•
•
•
Data Block n-1
Data Block n
-TM-
EOF1
EOF2
Data Set Trailer Labels
-TM-
-TM-

Tape Media

❒ **The previous page describes the layout of records on an IBM standard labelled tape**

- ♦ **Note that for tape the larger the block size the better, in terms of using the media and usually in terms of performance**

- ♦ **Maximum block size for tape is 256KB, with design allowing blocksize up to 2GB in the future**

❒ **The newest technologies do not physically record data on tape this way, but they logically simulate this arrangement, so your program (and your JCL) cannot tell the difference**

❒ **The latest technologies involve**

- ♦ **Cartridges ("square tape") replacing traditional reels ("round tape")**

- ♦ **New generations of cartridges**

- ♦ **Hardware data compaction - effectively giving as much as 1600GB (1.6TB) per cartridge**

  - ✗ Selectable through JCL (**TRTCH=COMP** or **TRTCH=NOCOMP** on the DD statement at create time)

- ♦ **Automated (robotic) cartridge mounting and demounting**

- ♦ **Virtual Tape Servers (VTS) - using DASD to appear as tape volumes**

Data Set Naming Rules

❒ A qualifier is a string of 1-8 alphanumeric or national ($ # @) characters, the first of which is not numeric

❒ A data set name (DSNAME) consists of 1 or more qualifiers, separated by periods, up to a maximum of 44 characters

Examples:

```
MYFILE

SYS1.LINKLIB

$TRNCM.TRAIN.LIBRARY

DEPT56.PAYROLL.EXTRACT.JANUARY.TEMP#1
```

❒ We use the term "level" to indicate where in a data set name a qualifier is, and you'll hear expressions like

- ♦ High level qualifier (= leftmost qualifier)
- ♦ Low level qualifier (= rightmost qualifier)
- ♦ Fully-qualified data set name

Units

❐ **Recall that by "UNIT" we mean an indication if a file needs a tape drive or a disk drive to be read from or written to**

- ♦ **Essentially: is it a tape file or a disk file?**

  - ✗ There may or may not be an installation default UNIT, but it never hurts to specify explicitly at data set create time

❐ **There are several ways to specify a unit, but the most common by far is to use a group name (also called an esoteric name)**

  - ✗ Each installation can map a pool of devices to these names

  - ✗ As your configuration changes, simply change the mappings, not your JCL

Common Unit Group Names

❐ For every z/OS installation, these unit names are defined automatically:

- ♦ SYSALLDA - assigned to all disk drives on the system
- ♦ SYSSQ - assigned to all tape and disk drives on the system

❐ All other group names are installation chosen, but some typical group names are:

- ♦ SYSDA - like SYSALLDA; most installations support this
- ♦ TAPE - for some subset of tape drives
- ♦ CART - for cartridge tape drives
- ♦ TSOWK - pool of disk drives used to hold TSO work files

Volumes

❒ **Volumes are uniquely identifed by their volume serials**

- ♦ **Volume serials are 1-6 characters long**

- ♦ **Not necessarily numeric, for example:**

```
SPOOL1
SYSRES
111111
MICKEY
DOPEY
DAFFY
```

❒ **You do not normally specify a volume serial at create time**

- ♦ **For new tape files, the operator will be asked to mount a <u>scratch volume</u>**

- ♦ **For new disk files, the allocation routines go on a shopping expedition, looking at all eligible DASD volumes for one that has enough space to meet your needs**

❒ **Data sets are located, then, by their <u>unit type</u> and <u>volume serial</u>**

Catalogs

❒ **Contain a list of data set names the system is to keep track of**

❒ **Each data set name is associated with the unit type (*e.g.*: 3490, 3380, 3390) and the volume serial(s) that identify where the data set resides**

❒ **Catalogs are themselves kept on disk**

❒ **Note that a data set does not need to be cataloged (as far as z/OS is concerned), but it is recommended**

- ♦ **Data sets coming in on tape from outside your installation are, of course, not cataloged when they arrive**

Section Preview

❒ **Tape and DASD DD Statements**

- ♦ **Tape and Disk Data Sets, DD Statements**
- ♦ **Building Tape and DASD DD Statements**
- ♦ **Sample DD Statements**
- ♦ **Data Flow Diagrams**
- ♦ **JOB Using Tape And Disk Data Sets (Machine Exercise)**

Tape and Disk Data Sets, DD Statements

- **DD statements for tape and disk data sets, may include these allocation parameters:**

  - **DSNAME (or DSN) - the name of the data set**
  - **DISP - the status of the data set at the beginning of the step (existing or new, shared or exclusive allocation), and how the data set should be disposed of at the end of the step**
  - **UNIT - indicates the type of machine needed to access the media (tape drive or disk drive)**
  - **SPACE - for disk: how much space do you think you'll need initially (primary amount), and if you fill that up, how much additional space would you like (secondary amount)**
  - **VOL - indicates which volume (volume serial number) and possibly a request to keep a tape volume mounted on a tape drive for a later step in the job**
  - **LABEL - for tapes: which file on the tape; what type of label processing to perform**

- **DD statements for tape and disk may also include these DCB parameters for use by OPEN:**

  - **LRECL, BLKSIZE, RECFM for any data set**
  - **TRTCH for tape data sets**
  - **DSORG for disk data sets**

Building Tape and DASD DD Statements

//ddname **DD {DSNAME|DSN}=*full_dataset_name*,**

- **From the program documentation**

// DISP=(*start*,*normal_disp*,*abnormal_disp*),

start choices

- **NEW - data set is being created in this step; exclusive use**
- **OLD - update or overwrite existing data set; exclusive use**
- **MOD - if data set exists, update; else create; exclusive use**
- **SHR - not for tape; data set exists; read only; non-exclusive**

normal_disp choices

- **KEEP - leave as it is on entry to step: cataloged or uncataloged**
- **DELETE - remove entry from VTOC (DASD) and from catalog**
- **PASS - for later use in same job; allows finding without cataloging**
- **CATLG - keep the file and make an entry to catalog**
- **UNCATLG - remove entry from catalog but keep the data around**

abnormal_disp choices

- **KEEP - leave as it is on entry to step: cataloged or uncataloged**
- **DELETE - remove from VTOC (DASD) and from catalog**
- **CATLG - keep the file and make an entry to catalog**
- **UNCATLG - remove entry from catalog but keep the data around**

DISP - Some Samples

```
DISP=(NEW,DELETE,DELETE)    <— (This is the default DISP)
DISP=(NEW,DELETE,CATLG)
DISP=(NEW,CATLG,DELETE)
DISP=(NEW,KEEP,DELETE)
DISP=(,KEEP,DELETE)
DISP=(NEW,PASS,DELETE)
DISP=(,PASS,CATLG)
DISP=(MOD,DELETE,KEEP)
DISP=(MOD,DELETE,UNCATLG)
```

Rules

- ♦ **The default first disposition is NEW**
- ♦ **The second disposition is implied by the first**
  - ✗ **The status of the data set before the step should be the status of the data after the step ends, thus**
    - ➢ NEW implies DELETE
    - ➢ SHR, MOD and OLD imply KEEP, if data set was cataloged, it will remain cataloged; MOD implies delete if the data set did not exist
- ♦ **The ABEND disposition defaults to whatever the second disposition is**
  - ✗ Except PASS implies DELETE for new data sets and KEEP for existing data sets

DISP - Quiz

❒ **What is the full disposition in the following cases?**

DISP=(,KEEP)

DISP=(NEW,,CATLG)

DISP=MOD

DISP=(MOD,PASS)

DISP=OLD

DISP=SHR

Notes - caution

- ♦ **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**

- ♦ **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

  ✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❐ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	
DISP=MOD	
DISP=(MOD,PASS)	
DISP=OLD	
DISP=SHR	

Notes - caution

- **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**
- **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❐ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	**NEW,DELETE,CATLG**
DISP=MOD	
DISP=(MOD,PASS)	
DISP=OLD	
DISP=SHR	

Notes - caution

- ♦ **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**
- ♦ **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❐ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	**NEW,DELETE,CATLG**
DISP=MOD	**MOD,{DELETE|KEEP},{DELETE|KEEP} - depending**
DISP=(MOD,PASS)	
DISP=OLD	
DISP=SHR	

Notes - caution

- ♦ **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**

- ♦ **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

  ✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❒ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	**NEW,DELETE,CATLG**
DISP=MOD	**MOD,{DELETE|KEEP},{DELETE|KEEP} - depending**
DISP=(MOD,PASS)	**MOD,PASS,{DELETE|KEEP} - depending**
DISP=OLD	
DISP=SHR	

Notes - caution

- **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**

- **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❒ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	**NEW,DELETE,CATLG**
DISP=MOD	**MOD,{DELETE|KEEP},{DELETE|KEEP} - depending**
DISP=(MOD,PASS)	**MOD,PASS,{DELETE|KEEP} - depending**
DISP=OLD	**OLD,KEEP,KEEP**
DISP=SHR	

Notes - caution

♦ **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**

♦ **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

✗ **Remember, NEW is implied if you omit the DISP parameter!**

DISP - Quiz

❒ **What is the full disposition in the following cases?**

DISP=(,KEEP)	**NEW,KEEP,KEEP**
DISP=(NEW,,CATLG)	**NEW,DELETE,CATLG**
DISP=MOD	**MOD,{DELETE|KEEP},{DELETE|KEEP} - depending**
DISP=(MOD,PASS)	**MOD,PASS,{DELETE|KEEP} - depending**
DISP=OLD	**OLD,KEEP,KEEP**
DISP=SHR	**SHR,KEEP,KEEP**

Notes - caution

♦ **When any step in a job refers to a data set with DISP of NEW, MOD, or OLD, the system converts all references to that data set in that job to DISP of OLD (exclusive use)**

♦ **In some installations, if a data set is specified with a DISP of NEW, the system automatically deletes any data set with that name first!**

✗ **Remember, NEW is implied if you omit the DISP parameter!**

Building Tape and DASD DD Statements, continued

//   UNIT=(*dev_type*[,*count*]),

- ♦ *dev_type* **is unit name like we already discussed;** *count* **is how many devices of that type to allocate (default is one)**

- ♦ **UNIT is needed for new data sets and existing non-cataloged data sets**

- ♦ **UNIT should not be coded for existing cataloged data sets or data sets that are PASSed from a previous step**

Examples

```
//        UNIT=CTAPE

//        UNIT=(TAPE,2)
```

Building Tape and DASD DD Statements, continued

```
//          VOL=SER=xxxxxx,
                 or                               omitting VOL causes a
//          VOL=SER=(xxxxxx,xxxxxx,...)           non-specific request
                 or
//          VOL=(,RETAIN)      (tape only: keep volume mounted on unit)
```

- **VOL is short for VOLUME; SER is short for SERIAL**
- **VOL is needed for existing data sets that are not cataloged**
- **VOL is needed when stacking multiple files on a particular tape cartridge / reel**
- **VOL should not be coded for existing cataloged data sets (except possibly VOL=(,RETAIN))**
- **If VOL is omitted for new data sets:**
  - ✗ Tape: system requests a scratch tape
  - ✗ Disk: system goes on a search for a unit / volume with enough space to hold the amount requested
- **A single tape data set can extend across 255 volumes**

Examples

```
//        VOL=SER=MICKEY

//        VOL=(,RETAIN)

//        VOL=(,RETAIN,SER=BAMBI)
```

Building Tape and DASD DD Statements, continued

//   LABEL=(*file_no*,*type_of_label_processing*),

or SPACE and possibly AVGREC, discussed on next page

- **LABEL is for tape data sets**
  - ✗ Although the presence of LABEL does not preclude the need for the UNIT parameter

- *file_no* **is an integer, indicating which file on the tape this data set is (default: 1)**

- *type_of_label_processing* **is an alpha code; most common:**
  - ✗ SL: Standard Label processing (the default if code omitted)
  - ✗ NL: No Label processing (non-labeled tape)
  - ✗ BLP: Bypass Label Processing (typically for VSE tape)

- **LABEL must be coded when accessing stacked files on a tape after the first (unless the data set is cataloged or PASSed from an earlier step in the same job)**

Examples

```
//        LABEL=(1,SL)

//        LABEL=5
```

Building Tape and DASD DD Statements, continued

//   SPACE=(*size*,(*pri*[,*sec*[,*dir*]])[,RLSE])[,AVGREC={U|K|M}]

- **SPACE is required for new DASD data sets; AVGREC works with the SPACE parameter, but may be omitted**

- **SPACE and AVGREC are not required for existing files**

- *size* **is "the unit of space request" and is one of**

  TRK — request is for tracks
  CYL — request is for cylinders
  nnnnn **— request is for physical blocks 'nnnnn' bytes long**
  • *nnnn* = 1 to 65535
  <u>or</u> — request is for logical records '*nnnnn*' bytes long (only if AVGREC is also coded or implied by DATACLAS)

- *pri* **says how many chunks of size** *size* **should be allocated initially (primary allocation; maximum of 7 digit number)**

- *sec* **says how many chunks of size** *size* **should be allocated for additional space as needed (secondary allocation; maximum of 7 digit number)**

- *dir* **says how many directory blocks to put at front of a PDS**

- **RLSE says: when the step completes normally, release any unused space from the allocated amount**

DD Statements: SPACE

- ♦ **For AVGREC, 'M' says *pri* and *sec* should be multiplied by a million, 'K' says *pri* and *sec* should be multiplied by a thousand, and 'U' says take *pri* and *sec* exactly as specified**

Sample SPACE parameters:

```
SPACE=(TRK,(10,15))
SPACE=(CYL,5)
SPACE=(1024,(100,200))
SPACE=(TRK,(5,2,12))
SPACE=(CYL,(15,10),RLSE)
SPACE=(CYL,(5,2,30))
```

❒ **Note that you will get one primary allocation and up to 15 secondary allocations on a single volume**

- ♦ **However, extended sequential, PDSE, and VSAM data sets can have up to 123 allocations on a single volume**

DD Statements: AVGREC

AVGREC	**—**	**Number of records represented in the primary and secondary values of the SPACE parameter: how many records; possible values:**

U - units
K - thousands
M - millions

Example

- **If SPACE and AVGREC are coded as:**

```
SPACE=(440,(100,20)),AVGREC=K
```

- **Then the space request is for a primary amount of 100,000 records, 440 bytes each, and a secondary amount of 20,000 records**

- **If AVGREC were omitted, this space request would be for 100 blocks primary and 20 blocks secondary, with a block size of 440 bytes**

❐ **Note that it is the presence or absence of AVGREC that determines if a numeric SPACE allocation amount is a record size or block size, respectively**

❐ **Note also that AVGREC only works if SMS is installed and active**

DD Statements: Date Protection

- ❒ **You have the ability to specify an expiration date for a new data set (usually tape, but it works for DASD also)**
  - ♦ **One of two formats**
    - ✗ EXPDT=yyyy/ddd — Expiration date, Julian format
    - ✗ RETPD=dddd — Retention period, dddd days
      - ➢ dddd may be up to 93000 (z/OS 1.10-1.12), but if a value is > 9999, then 9999 is used
      - ➢ RETPD value added to creation date to calculate expiration date
        - ➤ if calculated expiration date is > 12/31/2155, then 12/31/2155 is used
      - ➢ In z/OS 1.13, values up to 93000 are accepted and used, however the largest supported expiration date is still 12/31/2155

Samples

```
//        EXPDT=2012/174
//        RETPD=60
```

- ♦ **If a job or online user attempts to delete or update a date-protected data set, and the date protection has not expired, the operator is prompted to see if it is OK**
  - ✗ If the operator says it's OK, then the delete / update is allowed
- ♦ **Certain dates have special meanings; in particular, 99365, 99366, 1999/365, and 1999/366 are considered to mean "never expire"**
  - ✗ Also, some tape management software uses 99365 as an indication that this data set is not managed

Building Tape and DASD DD Statements, continued

❒ **So, you construct a DD statement for a tape or disk file using these parameters**

- **Including a parameter or not, based on the notes on the preceding pages**

```
//ddname      DD    {DSNAME|DSN}=full_dataset_name,

//            DISP=(start,normal_disp,abnormal_disp),

//            UNIT=(dev_type[,count]),

//            VOL=SER=xxxxxx,
                  or
//            VOL=SER=(xxxxxx,xxxxxx,...)
                  or
//            VOL=(,RETAIN)

//            LABEL=(file_no,type_of_label_processing),

//            SPACE=(size,(pri[,sec[,dir]])[,RLSE]),

//            AVGREC={U|K|M}

//            RETPD=dddd
                  or
//            EXPDT=yyyyddd
```

❒ **In addition, you may on occasion need to code various DCB parameters ...**

DCB Parameters

❒ **As mentioned earlier, you may need to code DCB parameters on a DD statement, if the necessary information cannot be obtained from the program or the label of the data set**

- ♦ **See especially page 377**

Summary / Reminder

- ♦ **LRECL - size of logical record in bytes**

- ♦ **BLKSIZE - size of physical block, in bytes**

  - ✗ NOTE: coding BLKSIZE=0 for a new data set requests the system to determine the block size (you must also specify LRECL and RECFM)

  - ✗ NOTE: BLKSIZE may be coded using a suffix of K, M, or G (for example: 12K, 52M, 1G); maximum value for non-tape is generally 32K, for tape 2G (although you can't really get a buffer that large)

- ♦ **RECFM - record format; one of: F, FB, V, VB, U**

- ♦ **DSORG - data set organization; PO for Partitioned Organization (PDS), PS for Physical Sequential**

- ♦ **TRTCH - tape recording technique; COMP or NOCOMP for compaction or not, respectively**

Some Typical DD Statements

```
//TAPEOUT   DD  DSN=D55.GOODTRNS,DISP=(,CATLG),UNIT=TAPE


//OUT2      DD  DSN=D55.ISD44.INTERIM,DISP=(,PASS),
//              UNIT=TAPE,VOL=(,RETAIN),BLKSIZE=0


//TAPEIN    DD  DSN=D55.CUST.MASTER,DISP=OLD


//IN2       DD  DSN=EAGLE.YRTRANS,DISP=MOD,VOL=(,RETAIN)


//STACKOUT  DD  DSN=SLAYER.UPDTS,DISP=(,CATLG),UNIT=TAPE,
//              LABEL=5,VOL=(,RETAIN,SER=BAMBI),
//              TRTCH=COMP


//SOUT2   DD  DSN=DOALL.HOLD,DISP=(,CATLG),UNIT=(TAPE,2),
//            RETPD=60


//STEPLIB  DD  DSN=DEPT22.LOADLIB,DISP=SHR


//MOUT     DD  DSN=OURS.NEW.MASTER,DISP=(,CATLG),
//             UNIT=SYSDA,SPACE=(CYL,(12,4))


//THEFILE  DD  DSN=D345T.DSMNS.PERSNL,DISP=(,CATLG),
//             UNIT=SYSDA,SPACE=(420,(100,20)),AVGREC=K
```

Member Names and Library Names

❒ **If the DSN on a DD statement is the name of a library (PDS), the library is assumed to be available as a place to find programs or data (input), or a place to put programs or data (output)**

input:

```
//STEPLIB  DD DSN=$TRNCM.TRAIN.LOADLIB,DISP=SHR
```

output:

```
//SYSUT5   DD  DSN=PAYROLL.SYMBOLS.LIB,DISP=OLD
```

❒ **If the DSN on a DD statement is the name of a library followed by a membername in parentheses, the member is treated as a sequential file itself (input), or as the name of a new member in the library (output)**

- ♦ **For output, if the named member already exists, it will be replaced**

input:

```
//SYSIN    DD  DSN=BIGCO.SRCE.COBOL(CALCWIT),DISP=SHR
```

output:

```
//SYSLMOD  DD  DSN=CUST.PROD.LODLIB(SMREPT),DISP=OLD
```

❒ **Warning: If you specify DISP=(OLD,DELETE) when a member is specified, at the end of the step, the system deletes the library, not the member!**

DD DUMMY

❒ **If you specify 'DUMMY' as the first operand (it is positional) on a DD statement, the system processes the file differently:**

- **Allocation does not try to find a DUMMY data set**
- **An attempt by the program to READ to a DUMMY file raises the end of file condition immediately**
- **An attempt by the program to WRITE to a DUMMY file is ignored**
- **Deallocation does not try to KEEP, DELETE, or CATALOG a DUMMY data set**

Example

```
//MYFILE   DD  DUMMY,DSN=D53.GOODTRANS,
//   LRECL=240,RECFM=FB
```

❒ **This is primarily used for program testing; a few programs check for the presence of a DUMMY data set**

Notes:

- **If DCB information is not complete in the program, it will have to be supplied on the DD statement for a file coded as 'DUMMY', otherwise, OPEN will fail (see example above)**
- **Applies to sequential and VSAM data sets**
- **Alternatively specify a DUMMY data set by coding "DSN=NULLFILE"**

Data Flow Diagrams

❐ **Jumping from a textual / verbal description of a task to JCL can sometimes be tricky**

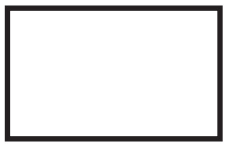
❐ **One technique that might be helpful is Data Flow Diagrams**

- ♦ **Convert the description of the task at hand to a diagram**
- ♦ **Then convert the diagram to JCL**

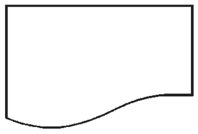
❐ **Basically, you use a small set of symbols to represent programs and data in a job, then show flow of data through connectives (pointed lines)**

Data Flow Diagrams, continued

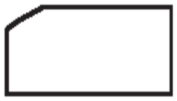
❒ **The figures to use are:**



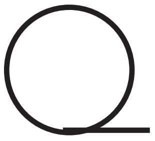
♦ **Box - to represent a program**



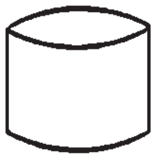
♦ **Form - to represent a report (sysout-type data)**



♦ **Cards - to represent sysin-type data**



♦ **Tape - to represent a tape reel or cartridge**



♦ **Disk - to represent a disk file**

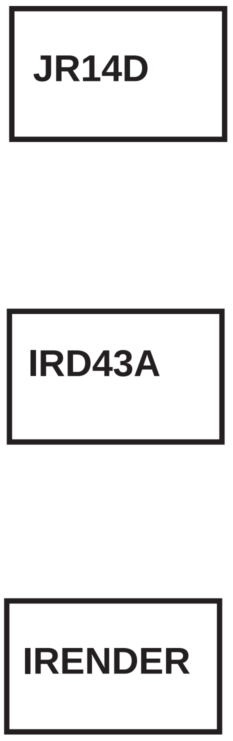
Data Flow Diagrams, continued

❒ **The process, then, is this**

- **From the description of the job, draw a box for each step; label each box with the program to be run at that step**

  ✗ Note that the boxes are drawn one above the other, in the order the programs (steps) are to run

  ✗ For example, you might show a three step job this way:

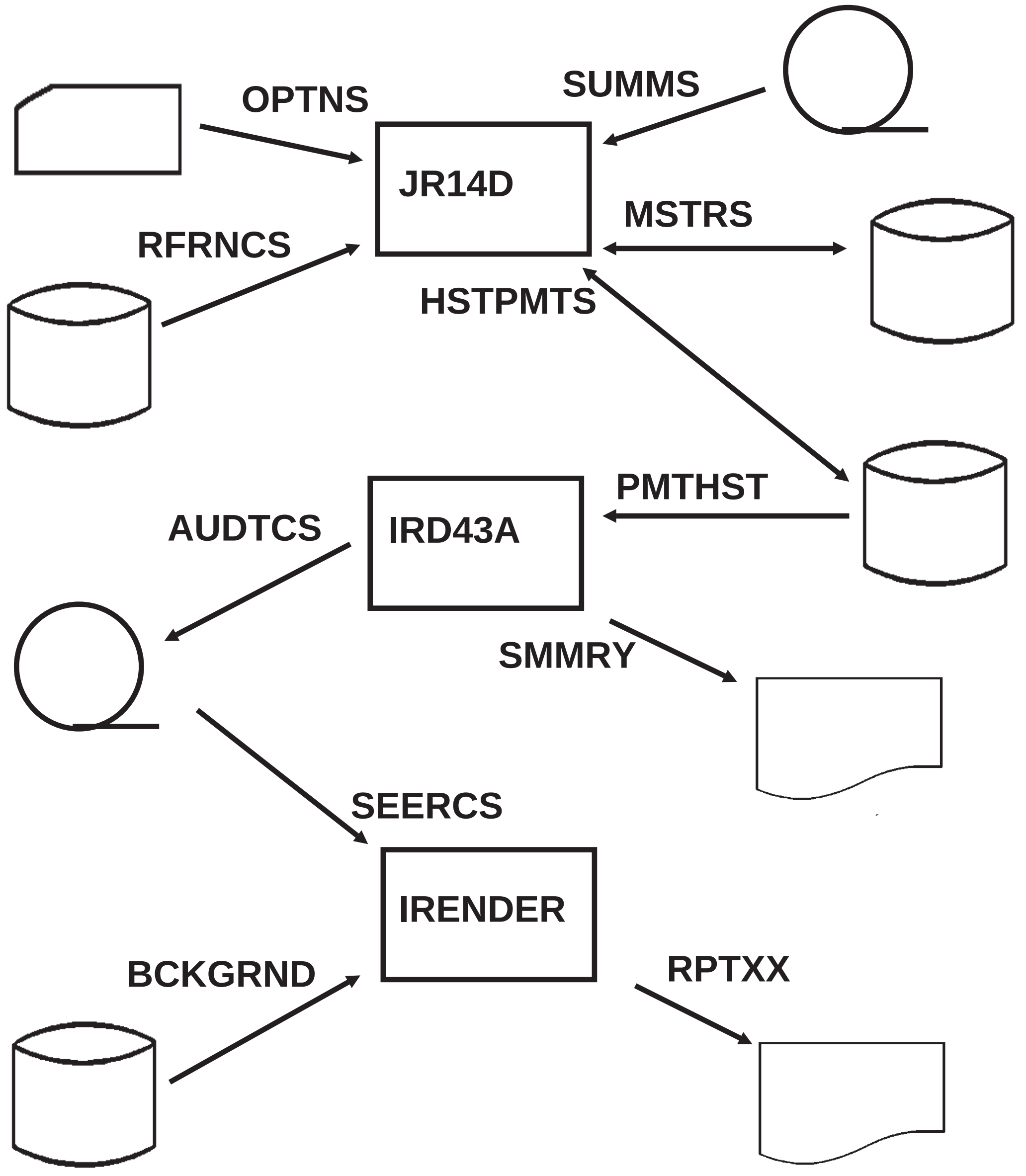


Data Flow Diagrams, continued

❒ **The process, continued**

- ♦ **Near each box, sketch a figure for each file the program needs, and label the figure with the DDname the program uses to reference the file**

  ✗ Draw an arrow between the figure and the box to show input, output, or update as appropriate



Data Flow Diagrams, continued

<u>Notes</u>

♦ **Place DDname labels as close to the arrows as possible and close to the box representing the program using the files**

♦ **Note that a file might be used by multiple programs**

✗ Each step will use it's own DDname, and you will need to code an appropriate DD statement for the file in each step it is used

❐ **Then, you can use the data flow diagram to code the initial JCL skeleton: program names on EXEC statements and DD names on DD statements; step order set; DD statements tied to the correct step:**

```
//-----     JOB   ------------------------------
//STEP1    EXEC  PGM=JR14D
//OPTNS     DD
//SUMMS     DD
//RFRNCS    DD
//MSTRS     DD
//HSTPMTS   DD
//STEP2    EXEC  PGM=IRD43A
//PMTHST    DD
//AUDTCS    DD
//SMMRY     DD
//STEP3    EXEC  PGM=IRENDER
//SEERCS    DD
//BCKGRND   DD
//RPTXX     DD
```

Data Flow Diagrams, continued

- ❐ Next, you might add some SYSUDUMP or CEEDUMP DD statements and any necessary STEPLIB DD statements
  - ♦ These often don't show up in data flow diagrams

Resulting in, so far:

```
//-----     JOB   ------------------------------
//STEP1    EXEC  PGM=JR14D
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//OPTNS     DD
//SUMMS     DD
//RFRNCS    DD
//MSTRS     DD
//HSTPMTS   DD
//SYSUDUMP  DD
//STEP2    EXEC  PGM=IRD43A
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//PMTHST    DD
//AUDTCS    DD
//SMMRY     DD
//SYSUDUMP  DD
//STEP3    EXEC  PGM=IRENDER
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//SEERCS    DD
//BCKGRND   DD
//RPTXX     DD
//SYSUDUMP  DD
```

Data Flow Diagrams, continued

- For sysin-type (card image) files, we know we can code '*' for the operand

- For sysout-type (report) files, we know we can code "SYSOUT=*" for the operand (you might have additional operands, of course)

- For tape and disk files, we can code DISP parameters based on the arrows

We now have this much:

```
//-----      JOB   ------------------------------
//STEP1    EXEC  PGM=JR14D
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//OPTNS     DD   *
//SUMMS     DD   DISP=OLD
//RFRNCS    DD   DISP=SHR
//MSTRS     DD   DISP=OLD
//HSTPMTS   DD   DISP=OLD
//SYSUDUMP  DD   SYSOUT=*
//STEP2    EXEC  PGM=IRD43A
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//PMTHST    DD   DISP=SHR
//AUDTCS    DD   DISP=(,CATLG)
//SMMRY     DD   SYSOUT=*
//SYSUDUMP  DD   SYSOUT=*
//STEP3    EXEC  PGM=IRENDER
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//SEERCS    DD   DISP=OLD
//BCKGRND   DD   DISP=SHR
//RPTXX     DD   SYSOUT=*
//SYSUDUMP  DD   SYSOUT=*
```

Data Flow Diagrams, continued

❒ **Other work grows from this**

- **For tape and disk files, we know we need to specify DSN= ...**

```
//-----     JOB   ------------------------------
//STEP1    EXEC  PGM=JR14D
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//OPTNS     DD   *
//SUMMS     DD   DISP=OLD,DSN=DEPT23.LOGS
//RFRNCS    DD   DISP=SHR,DSN=DEPT23.REFS
//MSTRS     DD   DISP=OLD,DSN=DEPT23.MASTER.FILE
//HSTPMTS   DD   DISP=OLD,DSN=DEPT23.HPMTS
//SYSUDUMP  DD   SYSOUT=*
//STEP2    EXEC  PGM=IRD43A
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//PMTHST    DD   DISP=SHR,DSN=DEPT23.HPMTS
//AUDTCS    DD   DISP=(,CATLG),DSN=DEPT23.AUDTCS
//SMMRY     DD   SYSOUT=*
//SYSUDUMP  DD   SYSOUT=*
//STEP3    EXEC  PGM=IRENDER
//STEPLIB   DD   DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//SEERCS    DD   DISP=OLD,DSN=DEPT23.AUDTCS
//BCKGRND   DD   DISP=SHR,DSN=DEPT23.BACK.GROUND
//RPTXX     DD   SYSOUT=*
//SYSUDUMP  DD   SYSOUT=*
```

- **For new files, we need to specify UNIT and VOL, explicitly or implicitly, and, if UNIT is disk-like, specify SPACE=**

- **Other clues from the description let you fill in the remaining blanks**

❒ **Eventually you fill in the entire jobstream and you're ready to test**

❒ **This isn't anything magic, but it may help you get started if you are having difficulty**

Exercise: JOB Using Tape And Disk Data Sets

Code the Job Control Language statements necessary to execute the job described below. The job is designed to produce a series of reports from a large inventory file.

Take some time and plan the JCL: break it down into steps and then break each step down into its component parts. Don't over-think on this exercise: if you are unsure about some point, do your best then go on. Later you can come back to the difficult point.

Test this job by running it with TYPRUN=SCAN in the JOB statement (this is because the programs and files don't really exist; we are just testing syntax here). So, start by creating a new member, JCLEX03, in your TR.CNTL data set, copy in JOB statement, and add the TYPRUN parameter to this statement (NOTE THE SPELLING: NO 'E').

Step One

Code an EXEC statement to run a program called ISDRUS01.

Code a DD statement using a DDname of SPECS. This is for reading some control statements, as card-images in the input stream, that will specify report criteria (for this lab, don't worry about the format or content of the control statements).

Code a DD statement with a DDname of MASTER to point to an inventory data set (a cataloged file named 'STORES.BASEREC.MASTER'). The program will read this file and select which records are to be used for the reports, based on the criteria specified in the control statements.

Selected records are written out to a tape file (standard labeled but only used for the duration of this job). For this tape data set, code a DD statement using a DDname of ABSTRACT and a data set name of 'STORES.SELECTED.RECORDS'.

ISDRUS01 also creates an edit listing that lists the control statements and any errors detected. For this edit listing, code a DD statement with a DDname of EDITLIST.

Exercise: JOB Using Tape And Disk Data Sets, continued

Step Two

Code an EXEC statement to invoke a program called SORT, and code DD statements based on this information:

The Sort reads some control statements that describe how a file is to be sorted, using a DD name of SYSIN...

For this step, the Sort control statements are stored as a member (named ISDSEQ01) in a cataloged PDS named APPLIC.PARMLIB.

Then, the Sort reads the file to be sorted (using a DD name of SORTIN)...

For this step, the file to sort is the tape file created in Step One above

SORT then sorts the input file, and writes the results to an output file (using a DDname of SORTOUT)...

For this step, the sorted file is to be written out to a temporary disk data set (usually, about 23,000 records have been extracted for reporting on; however, sometimes as many as 100,000 records have been selected). Use a data set name of 'STORES.SORTED.RECORDS'. Records are 480 bytes long.

During its run, Sort produces a listing of the control statements and some informational messages, using a DD name of SYSOUT...

For this step, route this listing to a print file.

Step Three

Invoke a program called ISDRPT03 to list the data set created from the SORT step. This program uses a DD statement with a DDname of ACT to reference this data set (STORES.SORTED.RECORDS), and a DD statement with a DDname of ISDRPTL4 for the report, which should go to a SYSOUT class of E.

This page intentionally left almost blank.

Clues for Writing JCL, continued

DD Statements

//ddname   DD   parameters

- ♦ **"parameters" - continued**
  - ✗ If a file is described as "a cataloged data set", code: DSN=data_set_name,DISP={SHR|OLD|MOD}; end of this statement
    - ➢ Code DISP=SHR if the data set is only being read; code DISP=OLD if the data set may be updated; code DISP=MOD if the data set is being extended
    - ➢ Note that you cannot tell from JCL alone if a cataloged data set is tape or disk

❒ **Notice: SYSIN and SYSOUT data sets do not have data set names; temporary data sets do not need to have names, although many installations recommend you name them**

❒ **Data set names are made up of qualifiers (1-8 alphanumeric or national characters, the first of which is not numeric) separated by periods up to a maximum of 44 characters**

- ♦ **Only the first 17 characters of a DSN are actually stored in the label, for tape data sets**
- ♦ **Normally follow installation naming convention standards**
- ♦ **For class exercises, use your TSO id as the high level (first) qualifier**

Clues for Writing JCL, continued

DD Statements

//ddname DD parameters

- **"parameters" - continued**

  ✗ If a file is described as "a new, tape data set", code the following, based on other available information:

  - ➢ DSN=data_set_name - the name of the data set
  - ➢ DISP=(NEW,**norm**,**abnorm**) - data set does not exist at beginning of step; for **norm**, indicate disposition of data set at normal end of step: PASS, DELETE, KEEP, CATLG, UNCATLG; for **abnorm**, indicate disposition of data set if the step "blows up": DELETE, KEEP, CATLG, UNCATLG
  - ➢ UNIT=unit_type - installation specific name for tape device (e.g.: UNIT=TAPE); code UNIT=(unit,2) for a multi-volume tape data set where you want to overlap rewinding of the first volume with writing to the second volume
  - ➢ LABEL=n - 'n' indicates file is nth file on the tape; default is '1'; if special label processing is required, may code LABEL=(n,{NL|BLP}), but this is rare
  - ➢ VOL=SER=ssssss - to specify a particular volume (usually only do this if coding LABEL=n where 'n' is 2 or more); VOL=(,RETAIN) if a volume is used later in the same job; VOL=(,RETAIN,SER=ssssss) to specify a particular volume that will be used later in the job; or omit this parameter
  - ➢ Code DCB parameters if necessary, as directed: LRECL=, RECFM=, BLKSIZE=, TRTCH=[NO]COMP

Clues for Writing JCL, continued

DD Statements

//ddname DD parameters

- **“parameters” - continued**
  - ✗ If a file is described as “an existing, tape data set”, and the data set is not cataloged, code the following, based on other available information:
    - ➢ DSN=data_set_name - the name of the data set
    - ➢ DISP=({OLD|MOD},**norm**,**abnorm**) - OLD implies data set exists at beginning of step, MOD implies it may or may not exist at beginning of step; for **norm**, indicate disposition of data set at normal end of step: PASS, DELETE, KEEP, CATLG; for **abnorm**, indicate disposition of data set if the step “blows up”: DELETE, KEEP, CATLG
    - ➢ UNIT=unit_type - installation specific name for tape device (e.g.: UNIT=TAPE); code UNIT=(unit,2) for a multi-volume tape data set where you want to overlap rewinding of the first volume with reading from the second volume
    - ➢ LABEL=n - 'n' indicates file is nth file on the tape; default is '1'; if special label processing is required, may code LABEL=(n,{NL|BLP}), but this is rare
    - ➢ VOL=SER=ssssss - to specify a particular volume (must do this for non-cataloged data sets); VOL=(,RETAIN,SER=ssssss) to specify a particular volume that will be used later in the job; or omit this parameter
    - ➢ DCB parameters: for standard labeled tape, may need RECFM, BLKSIZE, LRECL

Clues for Writing JCL, continued

DD Statements

//ddname   DD   parameters

♦ **“parameters” - continued**

✗ If a file is described as “a new, disk data set”, code the following, based on other available information:

➢ DSN=data_set_name - the name of the data set

➢ DISP=(NEW,**norm**,**abnorm**) - data set does not exist at beginning of step; for **norm**, indicate disposition of data set at normal end of step: PASS, DELETE, KEEP, CATLG, UNCATLG; for **abnorm**, indicate disposition of data set if the step “blows up”: DELETE, KEEP, CATLG, UNCATLG

➢ UNIT=unit_type - installation specific name for disk device (e.g.: UNIT=SYSDA)

➢ VOL=SER=ssssss - to specify a particular volume (rare for disk)

➢ SPACE=(units,(pri,sec)) - for new disk data set, specify amount of space needed; 'units' is 'TRK', 'CYL' or a number that signifies block size; 'pri' is a number indicating how many 'units' should be allocated for the primary (initial) request and 'sec' is a number indicating how many units should be allocated for any secondary (subsequent) requests; might also code: SPACE=(units,(pri,sec),RLSE) to release unused space

➢ AVGREC={U|K|M} - with SPACE, says 'units' indicates record size and 'pri' and 'sec' are numbers of records

➢ Code DCB parameters if necessary, as directed: LRECL, RECFM, BLKSIZE

Clues for Writing JCL, continued

DD Statements

//ddname   DD   parameters

- ♦ **"parameters" - continued**

  ✗ If a file is described as "an existing disk data set", it should also be cataloged: treat the data set as on page 5, code:

  ➢ DSN=data_set_name - the name of the data set

  ➢ DISP=({SHR|OLD|MOD},**norm**,**abnorm**) - for the first parameter, code DISP=SHR if the data set is only being read; code DISP=OLD if the data set may be updated; code DISP=MOD if the data set is being extended; for **norm**, indicate disposition of data set at normal end of step: PASS, DELETE, KEEP, UNCATLG; for **abnorm**, indicate disposition of data set if the step "blows up": DELETE, KEEP, UNCATLG

  ➢ UNIT and VOL should not be coded for cataloged data sets

❒ **Note for SMS environments: UNIT is ignored; most parameters may be omitted; may code DATACLAS, STORCLAS, or MGMTCLASS or allow these to default from ACS routines**

Clues for Writing JCL, concluded

DD Statements - Special cases and general rules

- **Temporary data sets do not need data set names, or you can use a DSN of two ampersands followed by a single qualifier (*e.g.:* &&TFL)**

  ✗ To refer to such a data set later, you can use a referback:
  //ddname DD DISP=OLD,DSN=*.stepname.ddname

- **New data sets must have UNIT coded, except for SYSOUT data sets (unless an SMS default has been established)**

- **New disk data sets must have SPACE coded (unless an SMS default has been established)**

- **Data sets coded with DISP=MOD must have UNIT= specified if there is a chance the data set does not exist at the beginning of the step; and if the unit is a disk type, you must also code SPACE=...; (unless an SMS default has been established)**

Section Preview

❒ SMS - System Managed Storage

- ♦ STORCLAS
- ♦ DATACLAS
- ♦ MGMTCLAS
- ♦ ISMF
- ♦ Output DD Statements With SMS

SMS - Storage Management Subsystem (or System Managed Storage)

- ❒ **A collection of products available to work with z/OS**
  - ♦ **Base component is standard**
    - ✗ But not all features are included (optional, extra charge)

- ❒ **Basic philosophy: system should manage data, based on user-specified characteristics**
  - ♦ **<u>DATACLAS</u> — Allocation defaults / physical characteristics**
  - ♦ **<u>STORCLAS</u> — Performance requirements / unit types**
  - ♦ **<u>MGMTCLAS</u> — Migration, retention, and backup criteria**

- ❒ **Note: certain JCL parameters and features are only available if SMS is installed and active**
  - ♦ **SMS can be installed with a minimum, default configuration, in order to provide support for parameters such as AVGREC, LIKE, REFDD, etc.**
  - ♦ **DATACLAS, STORCLAS, MGMTCLAS parameters may only be used if they have been defined by the systems staff**

SMS, continued

- **When you create a new data set in a system with SMS ...**

  - **You may specify one or more of DATACLAS, STORCLAS, and MGMTCLAS on the file's DD statement**

  - **Or let Automatic Class Selection (ACS) routines assign these attributes for you**

    - ACS routines are installation provided routines that assign DATACLAS, STORCLAS, and / or MGMTCLAS based on data set name patterns and possibly other criteria

    - When a new data set is created, the ACS routines are automatically invoked

  - **Or use a combination of these techniques**

    - Let one or more of the classes be assigned by ACS and assign the other(s) explicitly on your DD statement

- **All SMS data sets are both cataloged and kept track of in SMS-related data bases**

  - **Once a data set is created, the system finds out the DATACLAS / STORCLAS / MGMTCLAS from these sources**

    - Don't need to specify on JCL for existing data sets

STORCLAS

❒ **STORCLAS groupings are determined by your system storage administrator**

❒ **When a STORCLAS is defined and named, it includes the following kinds of information:**

- **Performance objectives (millisecond response times)**
- **Read or write bias (or no bias)**
- **Dual copy of file to be maintained**
- **Space allocation requests should be available for all volumes in a multi-volume file (pre-allocate space)**
- **For tape, allocation might be directed to a particular tape library**

❒ **When a STORCLAS is assigned to a new data set, UNIT and VOLUME parameters may not need to be specified**

- **The storage class often implies the UNIT and the VOLUME can be non-specific**

❒ **When a new data set is <u>not</u> to be SMS-managed, you must specify UNIT and specify or imply VOLUME**

DATACLAS

❒ DATACLAS groupings are determined by your system storage administrator

❒ When a DATACLAS is defined and named, it contains values for the following parameters (among others):

LRECL	— Record length
RETPD	— Retention period
<u>or</u>	
EXPDT	— Expiration date
RECFM	— Record format (for Non-VSAM files)
<u>or</u>	
RECORG	— File organization, VSAM files only
VOLUME	— Volume count (maximum number of volumes a data set can span)
DSNTYPE	— 'PDS' (for standard PDS), 'LIBRARY' (for PDSE), 'EXTENDED' (for extended sequential), 'HFS' (for Hierarchical File System), blank (for sequential or VSAM), 'PIPE' (z/OS UNIX named pipe)
SPACE	— SPACE and AVGREC parameters; for new disk data sets, how much space to ask for on disk
other	— more parameters not discussed in this course

❒ For tape data sets, only EXPDT, RETPD, RECFM and LRECL apply

MGMTCLAS

- ❒ **MGMTCLAS groupings are determined by your system storage administrator**

- ❒ **When a MGMTCLAS is defined and named, it contains the following kinds of information:**

  - ♦ **Expiration / Retention periods**

    - ✗ When should data be automatically deleted (if ever)

  - ♦ **Migration criteria**

    - ✗ When should data migrate from primary storage to secondary storage (on DASD but in compressed form)

    - ✗ When should data be archived

  - ♦ **Backup criteria**

    - ✗ How often should data be backed up

    - ✗ How many versions should be maintained

    - ✗ How long should backed-up data be retained

SMS, continued

- ❐ A data set assigned to a storage class is, by definition, SMS-managed

- ❐ SMS-managed data sets must belong to some storage class

  - ♦ You must either explicitly request a STORCLAS value or the ACS routines must assign the data set to a storage class

- ❐ SMS-managed data sets may also belong to a data class; if one is not requested, the ACS routines or system defaults can assign one

- ❐ Non-SMS-managed data sets may use data class assignments to pick up attributes from the class

- ❐ Management classes are optional for SMS-managed data sets, and may not be used for non-SMS-managed data sets

- ❐ Note that when a new data set is allocated, if it is SMS-managed SMS will catalog the data set at allocation time instead of step deallocation

  - ♦ If the data set is deleted at step end, that is still done then

ISMF

❒ **The Interactive Storage Management Facility is an ISPF-based tool to let you define, change, and / or find out your installation's definitions of DATACLAS, STORCLAS, and MGMTCLAS**

- ♦ **Some shops restrict availability to this product**
    - ✗ Or at least restrict capabilities for change

❒ **In any event, ISMF helps you see how available classes are defined, and even what classes are assigned to existing data sets**

- ♦ **But it does not show or describe the actual ACS routines in use in your installation**

Output DD Statements With SMS

- ❒ **If your installation has implemented SMS management for tape data sets, for new tape data sets it may be sufficient to code something like:**

```
//TAPEFL   DD    DSN=------------,DISP=(,CATLG),UNIT=CTAPE
```

  - ♦ **UNIT is usually required since most shops will establish default unit type of disk, if any**
  - ♦ **VOLUME can be omitted, unless you want VOL=(,RETAIN)**
  - ♦ **A RETPD or EXPDT value may be coded, or it can also come from a specified or ACS-supplied DATACLAS, as can RECFM and LRECL**
  - ♦ **SMS-managed tape data sets must be SL**

- ❒ **From a JCL perspective, SMS is not a big deal for tape data sets, except it may be useful for**
  - ♦ **Managing tape libraries**
  - ♦ **Using DASD buffering to minimize mounts**
  - ♦ **Ensuring allocation to tape devices with compaction capability (or not)**

Output DD Statements With SMS, 2

❒ **If your installation has implemented SMS, for new disk files it may be sufficient to code something like this:**

```
//THEFLE     DD        DSN=------,DISP=(,CATLG)
```

- **UNIT can be deduced from an ACS-assigned STORCLAS or the default unit type established with SMS can be used**

  - ✗ The UNIT parameter for new disk files is supposedly ignored by SMS, but in many shops with SMS, UNIT may still be specified (and may even be required)

- **You can, of course, explicitly code DATACLAS, STORCLAS, and MGMTCLAS parameters**

- **You may omit or code VOLUME, depending on your needs**

- **As usual, code DCB related parameters only as needed**

- **SPACE is obtained from DATACLAS, although you may explicitly code the SPACE parameter and the AVGREC parameter**

❒ **So SMS can, in fact save you JCL coding time, once you know your installation's SMS classes**

- **Especially useful for DATACLAS attributes**

Section Preview

❐ **Looking at Job output**

❐ **Other DD techniques and parameters**

- ♦ **Temporary data sets**
- ♦ **Concatenation of DD statements**
- ♦ **NEWF2F (Machine Exercise)**

Job Output

- ❒ Output from a job is written to tape, disk, terminals, microfiche, optical output devices, and so on

  - ♦ And printers

- ❒ Printed output is called SYSOUT data

  - ♦ SYSOUT data may be kept on a display queue (TSO HELD) so that you may browse it under ISPF 3.8, SDSF, IOF, or other similar products

  - ♦ SYSOUT data may be routed directly to a printer

Job Output Organization

❒ **Although the format varies a little with every installation, the contents of a job's SYSOUT data generally follow this pattern:**

- ♦ **One or more banner pages, or separator pages (only produced when the data is actually printed; not seen when data is displayed at terminal)**
- ♦ **A JES JOB log, either JES2 or JES3 (JESMSGLG)**
- ♦ **The listing of the JCL that makes up the job (JESJCL)**
- ♦ **System messages - composed primarily of allocation and deallocation messages, as well as condition codes returned from each step (JESYSMSG)**
- ♦ **A separate listing for each SYSOUT file in the job (that is, one for each report)**

Sample Job Output

❐ **A sample JES2 job log:**

```
                        J E S 2  J O B  L O G  --  S Y S T E M
hh.mm.ss JOB03595  $HASP373 E0205141 STARTED - INIT   29 - CLASS
hh.mm.ss JOB03595  ACF9CCCD USERID E020514  IS ASSIGNED TO THIS
hh.mm.ss JOB03595  IEF403I E0205141 - STARTED - TIME=hh.mm.ss
hh.mm.ss JOB03595  +IN INIT ROUTINE
hh.mm.ss JOB03595  IEF404I E0205141 - ENDED - TIME=hh.mm.ss
hh.mm.ss JOB03595  $HASP395 E0205141 ENDED
------ JES2 JOB STATISTICS ------
  dd MMM yyyy JOB EXECUTION DATE
           cc CARDS READ
          ppp SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
           ss SYSOUT SPOOL KBYTES
         m.mm MINUTES EXECUTION TIME
```

❐ **If the system had asked for help from the operator for this job, these messages would show up here**

❐ **If a step abends, a symptom dump will appear here**

Sample Job Output, 2

❒ **A typical JCL listing might look like this**

```
 1 //E0205141 JOB (0,D127B),'DB2 TRAINING JOB',
   //          CLASS=H,TIME=(,10),MSGCLASS=T,NOTIFY=&SYSUI
   //*
   IEFC653I SUBSTITUTION JCL - (0,D127B),'DB2 TRAINING JOB',CLAS
 2 //STEP1    EXEC  PGM=OSWTO,PARM='IN INIT ROUTINE'
 3 //STEPLIB   DD   DISP=SHR,DSN=E020514.TR.LOAD
 4 //MASTER    DD   DISP=SHR,DSN=SYSS.TRAIN.INPUTA
 5 //MISTER    DD   DISP=(,CATLG),UNIT=SYSALLDA,
   //     DSN=E020514.NEW.INPUTA,SPACE=(TRK,2)
 6 //CARDSIN   DD   *
 7 //LISTING   DD   SYSOUT=*
 8 //PRINT    EXEC  PGM=NEWF2F
 9 //STEPLIB   DD   DISP=SHR,DSN=E020514.TR.LOAD
10 //INDD      DD   DISP=SHR,DSN=SYSS.TRAIN.INPUTA
11 //OUTDD     DD   SYSOUT=*
12 //OTHER     DD   DISP=(OLD,DELETE),DSN=E020514.NEW.INPUTA
```

❒ **Notice the statement numbers**

- ♦ **These are generated by the Converter, and are referenced in any diagnostic / informational messages**

❒ **If this job had invoked a cataloged procedure, all the statements from the procedure would also be listed here**

Sample Job Output, 3

❒ **A system messages listing might look like this**

```
ICH70001I IBMUSER  LAST ACCESS AT 13:47:50 ON THURSDAY, MAY 19,
IEF236I ALLOC. FOR E0205141 STEP1
IGD103I SMS ALLOCATED TO DDNAME STEPLIB
IEF237I 2178 ALLOCATED TO MASTER
IGD101I SMS ALLOCATED TO DDNAME (MISTER  )
        DSN (E020514.NEW.INPUTA                          )
        STORCLAS (STANDARD) MGMTCLAS (SEMIANN) DATACLAS (NODSORG
        VOL SER NOS= STSP05
IEF237I JES2 ALLOCATED TO CARDSIN
IEF237I JES2 ALLOCATED TO LISTING
IN INIT ROUTINE
IEF142I E0205141 STEP1 - STEP WAS EXECUTED - COND CODE 0000
IGD104I E020514.TR.LOAD                             RETAINED,  D
IEF285I   SYSS.TRAIN.INPUTA                            KEPT
IEF285I   VOL SER NOS= MVSP02.
IGD104I E020514.NEW.INPUTA                          RETAINED,  D
IEF285I   E020514.E0205141.JOB03595.D0000101.?         SYSIN
IEF285I   E020514.E0205141.JOB03595.D0000102.?         SYSOUT
IEF373I STEP/STEP1   /START yyyyddd.hhmm
IEF374I STEP/STEP1   /STOP  yyyyddd.hhss CPU    0MIN 00.01SEC SR
```

Sample Job Output, 4

❒ Sample system messages listing, continued

```
IEF236I ALLOC. FOR E0205141 PRINT
IGD103I SMS ALLOCATED TO DDNAME STEPLIB
IEF237I 2178 ALLOCATED TO INDD
IEF237I JES2 ALLOCATED TO OUTDD
IGD103I SMS ALLOCATED TO DDNAME OTHER
IEF142I E0205141 PRINT - STEP WAS EXECUTED - COND CODE 0000
IGD104I E020514.TR.LOAD                             RETAINED,  D
IEF285I   SYSS.TRAIN.INPUTA                            KEPT
IEF285I   VOL SER NOS= MVSP02.
IEF285I   E020514.E0205141.JOB03595.D0000103.?         SYSOUT
IGD105I E020514.NEW.INPUTA                          DELETED,  DD
IEF373I STEP/PRINT   /START yyyyddd.hhmm
IEF374I STEP/PRINT   /STOP  yyyyddd.hhmm CPU    0MIN ss.hhSEC SR
IEF375I  JOB/E0205141/START yyyyddd.hhmm
IEF376I  JOB/E0205141/STOP  yyyyddd.hhmm CPU    0MIN 00.04SEC SR
```

Sample Job Output, 5

❐ **Notice how this tracks**

- ♦ **Job Allocation (none in this example)**

First step: STEP1

- ♦ **Step Allocation**
- ♦ **Step Execution, including condition code**
- ♦ **Step Deallocation, including data set disposition and timing information**

Second step: PRINT

- ♦ **Step Allocation**
- ♦ **Step Execution, including condition code**
- ♦ **Step Deallocation, including data set disposition and timing information**

- ♦ **Job Deallocation and termination, including timing information**

This page intentionally left almost blank.

Temporary Data Sets

❒ **A data set that is created and deleted in the same job is called a <u>temporary data set</u>**

❒ **The operating system has a facility for generating names for temporary data sets for you, so you don't need to bother**

- ♦ **The down side is, you cannot KEEP or CATLG a data set with a temporary data set name**
- ♦ **Therefore, if you need to rerun or restart a job, temporary data sets created in the job are gone**
- ♦ **For this reason, many shops do not allow you to use data sets with temporary data set names in production**
  - ✗ Always catalog under a standard data set name, then add a step to delete temporary data sets at the end of job if everything went OK

Temporary Data Set Names

❒ **A temporary data set name is generated if you omit DSN entirely or if you code two ampersands followed by a single qualifier:**

```
//EDIT        EXEC     PGM=CHECKIT
//BADTRNS      DD    UNIT=SYSDA,DISP=(,PASS),SPACE=(1000,(20,10))
...
//STEP5       EXEC     PGM=ISPFLD
//TEMPHOLD     DD    DSN=&&WRK,UNIT=SYSALLDA,DISP=(,PASS)
...
//LASTEP      EXEC     PGM=CLEANERS
//SYSUT1       DD    UNIT=SYSALLDA,SPACE=(CYL,(2,2))
```

- ♦ **If the DISP parameter is (NEW,DELETE) (explicitly or implicitly), the data set is created and deleted in the same step**
- ♦ **If the DISP parameter is (NEW,PASS) (it is not allowed to be KEEP or CATLG for a temporary data set name), the data set is kept track of in the Passed Data Set Queue**
- ♦ **The third DISP subparameter is always treated as PASS**

❒ **To reference a PASSed temporary data set, either use a backwards reference or use the temporary DSN you gave it before:**

```
//INTRANS  DD   DSN=*.EDIT.BADTRNS,DISP=(OLD,DELETE)

//HELDTRN  DD   DSN=&&WRK,DISP=(OLD,PASS)

//OTHRWSE       DD   DSN=*.SYSUT1,DISP=(OLD,DELETE)
```

Using an Existing Data Set For a Model

❒ **If you are creating a data set that is to have the same characteristics as an existing data set, you have an additional choice (this only works if SMS is installed and active, although neither the new nor the original data sets need to be SMS-managed)**

- **LIKE=   name an existing cataloged data set, by its fully qualified name**

Notes

- **The new data set will have the following attributes copied from the existing data set: RECORG or RECFM, LRECL, KEYLEN, KEYOFF, DSNTYPE, SPACE, AVGREC**
  - ✗ For tape, only LRECL and RECFM are copied
  - ✗ The SPACE value that is used is the sum of the number of tracks for the first three extents, so override that if you need to
- **The existing data set cannot be a temporary data set**

Example

```
//NEWFILE  DD  DSN=SEQ2.INVEN.BACKUP,DISP=(,CATLG),
//    LIKE=SEQ2.INVEN
```

Concatenation

- You can concatenate multiple input files under a single DDname

  - The system will treat all the files as extensions of one another

  - For PDS's, the order of concatenation specifies the order of search

```
//JOBLIB        DD    DSN=ASPEN.TEST.LOADLIB,DISP=SHR
//              DD    DSN=DEPT057.PROD.ENVLIB,DISP=SHR
//              DD    DSN=SYS1.SCEERUN,DISP=SHR
```

  - For sequential data sets, when your program reads a record from a concatenated data set, all records are read from all files, in the order specified by the concatenation

```
//WEEKLYTR    DD        DSN=STORES.TRANS.MON,DISP=SHR
//            DD        DSN=STORES.TRANS.TUES,DISP=SHR
//            DD        DSN=STORES.TRANS.WED,DISP=SHR
//            DD        DSN=STORES.TRANS.THURS,DISP=SHR
//            DD        DSN=STORES.TRANS.FRI,DISP=SHR
```

Computer Exercise: NEWF2F

Code the JCL statements necessary to execute a three step job to copy an input data set several ways. Each step should execute the program named NEWF2F, which is found in the cataloged library named _________.TRAIN.LOADLIB. (Hint: put a STEPLIB in each step).

So, create a new member, JCLEX04, and copy in JOB as before; this time, we will run the job for real (that is, without TYPRUN= in the JOB statement). Next, code EXEC and DD statements using this information:

Step One

Run NEWF2F. It has these DD requirements:

A DD statement with the name INDD references the input data set, a cataloged data set with the name ________.TRAIN.INPUTA. (Informally, we'll call this data set just 'INPUTA'; records in INPUTA are 100 bytes long.) A DD statement with the name OUTDD creates a new, temporary disk data set (use LIKE and point to the input data set).

Step Two

Run NEWF2F again, only now,...

INDD should point to the new disk data set you created in Step One. OUTDD should describe another new temporary disk data set.

Step Three

Run NEWF2F a third time, and this time,...

INDD should point to the disk data set created in Step Two. OUTDD should direct the data set to a printer.

Exercise stretch: in Step One, concatenate INPUTX to INPUTA (INPUTX has the same data set name pattern, and the same record layout, as INPUTA).

Note

* Be sure to SHR INPUTA [and INPUTX if doing the stretch]!!

Section Preview

❐ **Utilities and Job Output Viewing**

- ♦ **Utilities**
- ♦ **IEFBR14**
- ♦ **IEBGENER**
- ♦ **IDCAMS**
- ♦ **SDSF | Flasher | IOF | (E)JES**
- ♦ **Utilities (Machine Exercise)**

Utilities

❒ In the programming business, a "utility" is a program that accomplishes a task that has to be done frequently by many people, such as:

- ♦ List files
- ♦ Backup and restore files
- ♦ Manage libraries
- ♦ Sort files
- ♦ Merge files

❒ IBM supplies a variety of utility packages and programs, and we discuss a number of them here

- ♦ Most shops also have a number of utility programs written in-house or purchased from "third party" software vendors (and we don't discuss any of these in this course)

The Non-utility Utility

❐ **A program that is often referred to as a utility is "IEFBR14"**

- **But IEFBR14 is not really a utility: the whole program is only four bytes long!**

- **IEFBR14 is a dummy program: all it does is "RETURN"**

- **You can take advantage of how allocation and deallocation work and use IEFBR14 merely as an excuse to pre-allocate or delete a non-VSAM data set:**

```
//ANY      EXEC    PGM=IEFBR14
//ANYFLE   DD   DSN=NEWMST.EDITED.CHECKED,DISP=(,CATLG),
//    UNIT=SYSDA,SPACE=(CYL,(10,2))
//OLDFLE   DD   DSN=OLDMST.USED.OBSOLETE,DISP=(OLD,DELETE)
//MAYBE    DD   DSN=MIGHT.BE.THERE,DISP=(MOD,DELETE,DELETE),
//    UNIT=SYSDA,SPACE=(CYL,5)
```

<u>When this step is run,</u>

- **Allocation creates space for the first file, locates the second file, and tries to locate the third file (if not found, allocates it)**
- **The program is run, returning promptly to the operating system**
- **Deallocation then catalogs the first file and deletes the second and third files!**
  - ✗ Recall that SMS-managed data sets are cataloged at allocation time, but the effect is the same

IEFBR14, continued

- **In using IEFBR14, you can code any DDname(s) you want, and as many DD statements as you need**

  - **And you can have as many IEFBR14 steps in a job as you need**

    - Delete one or more files at the beginning of a job

      - Use DISP=(OLD,DELETE) or even DISP=(MOD,DELETE)

    - Pre-allocate one or more files at the beginning of a job

      - Use DISP=(,PASS), DISP=(,KEEP), or DISP=(,CATLG)

    - Delete one or more files created earlier in the job

      - Use DISP=(OLD,DELETE)

  Caution

  - **When you allocate a new data set with IEFBR14, DCB information is not placed into the label, nor is an end of file indicator written in the data, until the data set is OPENed for output**

    - Opening the file for input before this is done can cause problems

    - However, if the data set is SMS managed, SMS will write an EOF indicator

IEBGENER - Generate Files

❒ **IEBGENER is a file-to-file-with-reformatting-ability utility(!)**

- ♦ **This utility uses the following DD statements**

  - ✗ SYSUT1 - source file; file used for input

  - ✗ SYSUT2 - target file; file used as output

  - ✗ SYSPRINT - utility messages file; describe problems or success

  - ✗ SYSIN - utility control statements; describe what reformatting is to be done

Copy existing files for backup or reblocking

- ✗ If SYSIN is specified as DUMMY, IEBGENER copies the SYSUT1 file to the SYSUT2 file with no changes in record format or content

  - ➢ If you don't supply any DCB information on the SYSUT2 file, IEBGENER uses the DCB information from the SYSUT1 file

  - ➢ But if you supply DCB information on the SYSUT2 DD statement, IEBGENER uses that information instead

- ✗ If SYSUT2 block size is not specified, it will be calculated by the System Determined Blocksize (SDB) routine; this might yield values greater than the previous maximum blocksize unless you code PARM='SDB={INPUT|SMALL}' on the EXEC statement that invokes IEBGENER; alternatively, PARM='SDB=LARGE' allows blocksize to be greater than 32760, which is supported for tape and dummy datasets

IEBGENER: File To File Example

```
//--------      JOB        --------
//COPYFLE       EXEC       PGM=IEBGENER,PARM='SDB=SMALL'
//SYSIN         DD         DUMMY
//SYSPRINT      DD         SYSOUT=*
//SYSUT1        DD    DSN=@DVELP.RGF098.INVNTRY,DISP=SHR
//SYSUT2        DD    DSN=@DVELP.INVNTRY.BACKUP,DISP=(,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(40,10),RLSE)
```

<u>A</u>ccess <u>M</u>ethod <u>S</u>ervices

- ❒ **The VSAM utility program**
- ❒ **Program name: IDCAMS**
  - ♦ **Batch program**
  - ♦ **TSO command processor**
- ❒ **Primary way to create / delete VSAM data sets**
  - ♦ **Cannot use JCL alone**
    - ✗ Except when using SMS (Storage Management Subsystem)
- ❒ **Provides many additional useful support functions**
- ❒ **Does work based on user <u>commands</u>**

Typical Access Method Services JCL

```
//--------      JOB     -----

//STEPX         EXEC    PGM=IDCAMS

//SYSPRINT      DD      SYSOUT=*

//SYSIN         DD      *
       .
       .     AMS command statements, such as
       .
     PRINT   INDATASET(VSM2.MIL.HRS.H2HI)   HEX
     DELETE   (VSM3.TABLES.LV009,   VSM2.TABLES.LV00A)
     DELETE   VSM.MM.LOMFRT.BANDIT
```

AMS Command Syntax

Syntax

COMMAND   parameters

- ❐ **Columns 2 - 72**

- ❐ **Parameters**

  - — **Positional**
  - — **Keyword**

    + Reserved word
    + Reserved word(value(s))

  - ♦ **Lists of values must be enclosed in parentheses, separated by commas and / or blanks**

  - ♦ **If there is only one element in the list, may omit the parentheses**

Examples

```
PRINT   INDATASET(VSM2.MIL.HRS.H2HI)   HEX    COUNT(50)

DELETE   (VSM3.TABLES.LV009,   VSM2.TABLES.LV00A)

DELETE   VSM.MM.LOMFRT.BANDIT
```

Abbreviations / Comments / Continuation

❒ **Most commands and parameters have abbreviations you may code instead of the entire reserved word:**

```
PRINT  INDATASET(...

PRINT  IDS(...
```

❒ **Comments:** **/* ..... */**

❒ **Continuation**

Use "–" or "+"

♦ **Just use "–" here; differences not worth discussing**

```
PRINT   IDS (VSM2.MIDORI.KEYS) –
        HEX –
        SKIP(31)  –
        COUNT(19)
```

♦ **Lack of a continuation character means end of a command**

❒ **Spaces may be freely inserted except in the middle of a value**

Using AMS To Process Non-VSAM Data Sets

❒ **Delete a cataloged non-VSAM data set:**

```
DELETE   dsname   [PURGE]   [NOSCRATCH]
```

Examples

```
DELETE   $TRNCM.TRAIN.INPUTH

DELETE   (PSHC.TRAIN.INPUTQ,   PSHC.TRAIN.PEOPLE)
```

❒ **Delete a member from a library (PDS or PDSE):**

```
DELETE   dsname(membername)
```

Example

```
DELETE   TFSHC.TRAIN.CNTL(WHOOPS)
```

Using AMS To Process Non-VSAM Data Sets, 2

❒ **Rename a data set, or a member of a library:**

```
ALTER   dsname   NEWNAME(newname)


ALTER   pdsname(membername)
           NEWNAME(pdsname(newmembername))
```

Examples

```
ALTER   JKIN.TRAIN.JUNK   NEWNAME(JKIN.TRAIN.GOOD)


ALTER   TFHHC.TEST.PLI(QEXER2)   —

           NEWNAME(TFHHC.TEST.PLI(QG17001))
```

Using AMS To Process Non-VSAM Data Sets, 3

❒ **Print all or some records:**

```
PRINT   IDS(dsname)   [CHAR | DUMP | HEX]
                 [SKIP(nnnn)]   [COUNT(mmmm)]
```

Examples

```
PRINT   IDS(TFJMM.MAYBE.FILE)   CHAR

PRINT   IDS(TS023.TRAIN.UNK)   SKIP(200)   COUNT(2)   HEX
```

❒ **Copy all or some records:**

```
REPRO   IDS(inputdsname)   ODS(outputdsname)
                 [SKIP(nnnn)]   [COUNT(mmmm)]
```

Example

```
REPRO   IDS(STNT329.TRAIN.DADA)   —

                 ODS(STNT222.TRAIN.DOODOO)
```

Using AMS To Process Non-VSAM Data Sets, 4

❒ **On all these examples, we have been referencing data sets by name: without JCL!**

For example:

```
//COPY            EXEC    PGM=IDCAMS
//SYSPRINT         DD     SYSOUT=*
//SYSIN            DD     *
     REPRO   IDS(ABSTR.SELECTED.RECS)   —
                ODS(ABSTR.BACKUP)
     PRINT   IDS(ABSTR.SELECTED.RECS)   CHAR
```

- **IDCAMS allows you to do this because it can issue dynamic allocation commands to allocate files on the fly**
  - ✗ In some ways this is cool (keeps the amount of JCL down)
  - ✗ But in other, subtle ways it is not a good idea (for example, dynamically allocated data sets are allocated with an implicit DISP of OLD, which keeps other users from accessing these files)

- **For REPRO and PRINT, you can also use standard allocation (JCL allocation) for the input and / or the output**
  - ✗ If the output is a new file, you must use standard allocation
  - ✗ Specify INFILE(*ddname*) for input and / or OUTFILE(*ddname*) for output
    - ➢ You must then include the appropriate DD statements in the step, of course (although you can make up your own DDnames) ...

Using AMS To Process Non-VSAM Data Sets, 5

Example Using Standard Allocation

```
//COPY           EXEC    PGM=IDCAMS
//SYSPRINT        DD     SYSOUT=*
//SYSIN           DD     *
     REPRO   INFILE(SOURCE)   OUTFILE(TARGET)
//SOURCE          DD     DSN=ABSTR.SELECTED.RECS,DISP=SHR
//TARGET          DD     DSN=ABSTR.BACKUP,DISP=(,CATLG),
//             LIKE=ABSTR.SELECTED.RECS
//PRINT          EXEC    PGM=IDCAMS
//SYSPRINT        DD     SYSOUT=*
//SAUCE           DD     DSN=ABSTR.SELECTED.RECS,DISP=SHR
//SYSIN           DD     *
     PRINT   INFILE(SAUCE)   CHAR
```

Or, equivalently

```
//COPY           EXEC    PGM=IDCAMS
//SYSPRINT        DD     SYSOUT=*
//SYSIN           DD     *
     REPRO   INFILE(SOURCE)   OUTFILE(TARGET)
     PRINT   INFILE(SOURCE)   CHAR
//SOURCE          DD     DSN=ABSTR.SELECTED.RECS,DISP=SHR
//TARGET          DD     DSN=ABSTR.BACKUP,DISP=(,CATLG),
//             LIKE=ABSTR.SELECTED.RECS
```

❒ **Note that the DD statements can go anywhere in the step (except in the middle of the SYSIN commands, of course)**

❒ **Also note that the output of PRINT goes to the SYSPRINT DD name by default, but you can specify OUTFILE for this, too**

System Display And Search Facility (SDSF)

Option S - maybe

❒ **An IBM program product, so not available at all installations**

Display Queues

ACTIVE	**—**	**TSO users logged on, jobs executing, started tasks**
INPUT	**—**	**Active jobs and jobs waiting for selection**
OUTPUT	**—**	**Non-held SYSOUT data**
HELD OUTPUT	**—**	**TSO held SYSOUT data**

Display Data Sets

SYSIN or SYSOUT

Display System Log

Control Jobs and SYSOUT Data

HOLD	**—**	**Jobs and SYSOUT data**
RELEASE	**—**	**Jobs and SYSOUT data**
CANCEL	**—**	**Jobs**
PURGE	**—**	**SYSOUT data**

❒ **Each user may or may not be authorized to perform any of these functions**

SDSF - Display Queues

❒ **From SDSF menu, select queue to display**

- ♦ **Enter character(s) on the command line (===>)**

DA	**—**	**Display Active queue**
I	**—**	**Display Input queue**
O	**—**	**Display Output queue (non-held SYSOUT data)**
H	**—**	**Display TSO Hold queue (TSO-held SYSOUT data)**
ST	**—**	**Display STatus of all jobs (input, running, output)**
LOG	**—**	**Display SYSLOG**

❒ **Each of the first five of these requests produces a scrollable list of jobnames**

- ♦ **Each of these displays has two columns we care about, labelled "NP" and "C"**

  - ✗ In the "NP" column, you may enter an "Action Character" next to the jobname you're interested in

  - ✗ The "C" column is used to specify a new JOB class or a new SYSOUT class, where appropriate

❒ **Note that which panels a user may see, which columns show on which panels, and which commands a user may enter are highly tailorable**

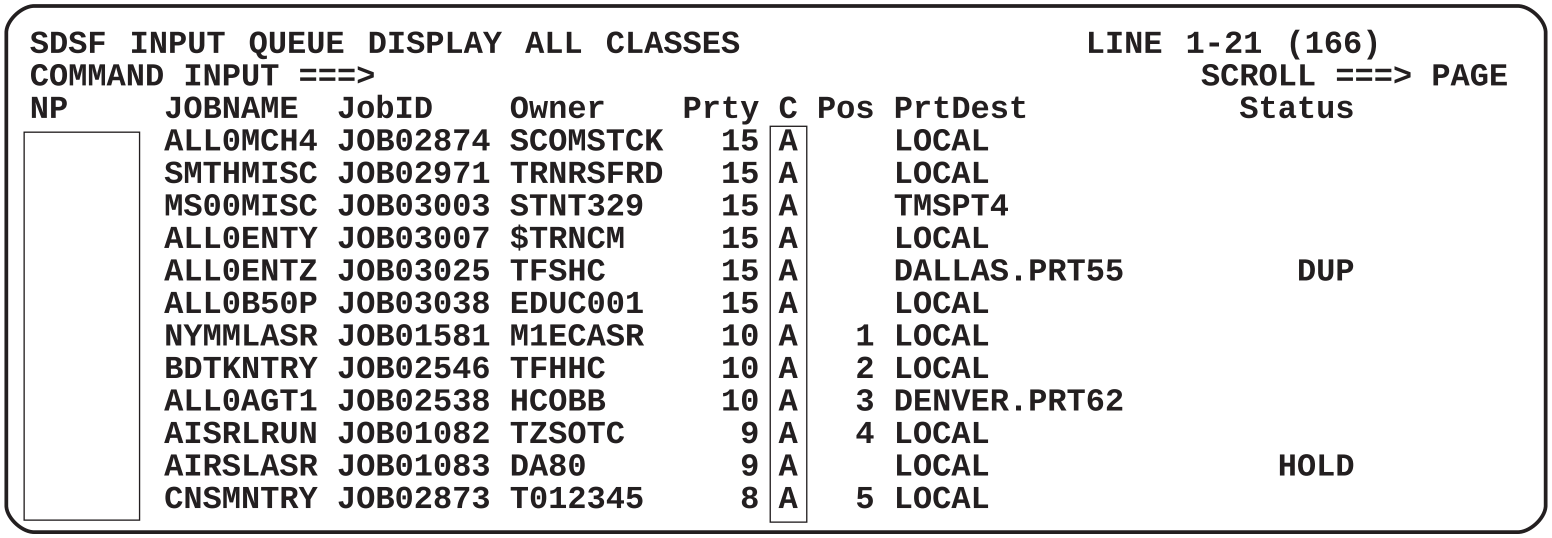
- ♦ **Your panels may not look just like those on the following pages, but the essentials are the same**

SDSF - Display Job Input Queues

❒ May request display of all jobs waiting to run (===> I) or only jobs in some specific job class(es) (===> IBCD)

- Up to seven classes may be specified

❒ Also, may ask to see only held jobs (===> I H) or only jobs that are not held (===> I NH)



```
SDSF INPUT QUEUE DISPLAY ALL CLASSES                 LINE 1-21 (166)
COMMAND INPUT ===>                                         SCROLL ===> PAGE
NP     JOBNAME  JobID    Owner    Prty C Pos PrtDest            Status
       ALL0MCH4 JOB02874 SCOMSTCK   15 A     LOCAL
       SMTHMISC JOB02971 TRNRSFRD   15 A     LOCAL
       MS00MISC JOB03003 STNT329    15 A     TMSPT4
       ALL0ENTY JOB03007 $TRNCM     15 A     LOCAL
       ALL0ENTZ JOB03025 TFSHC      15 A     DALLAS.PRT55          DUP
       ALL0B50P JOB03038 EDUC001    15 A     LOCAL
       NYMMLASR JOB01581 M1ECASR    10 A   1 LOCAL
       BDTKNTRY JOB02546 TFHHC      10 A   2 LOCAL
       ALL0AGT1 JOB02538 HCOBB      10 A   3 DENVER.PRT62
       AISRLRUN JOB01082 TZSOTC      9 A   4 LOCAL
       AIRSLASR JOB01083 DA80        9 A     LOCAL                HOLD
       CNSMNTRY JOB02873 T012345     8 A   5 LOCAL
```

'NP' action characters: H (hold), A (release), C (cancel), S (select to browse), SB (browse using ISPF Browse), SE (use ISPF edit), SJ (select submitted JCL using view)

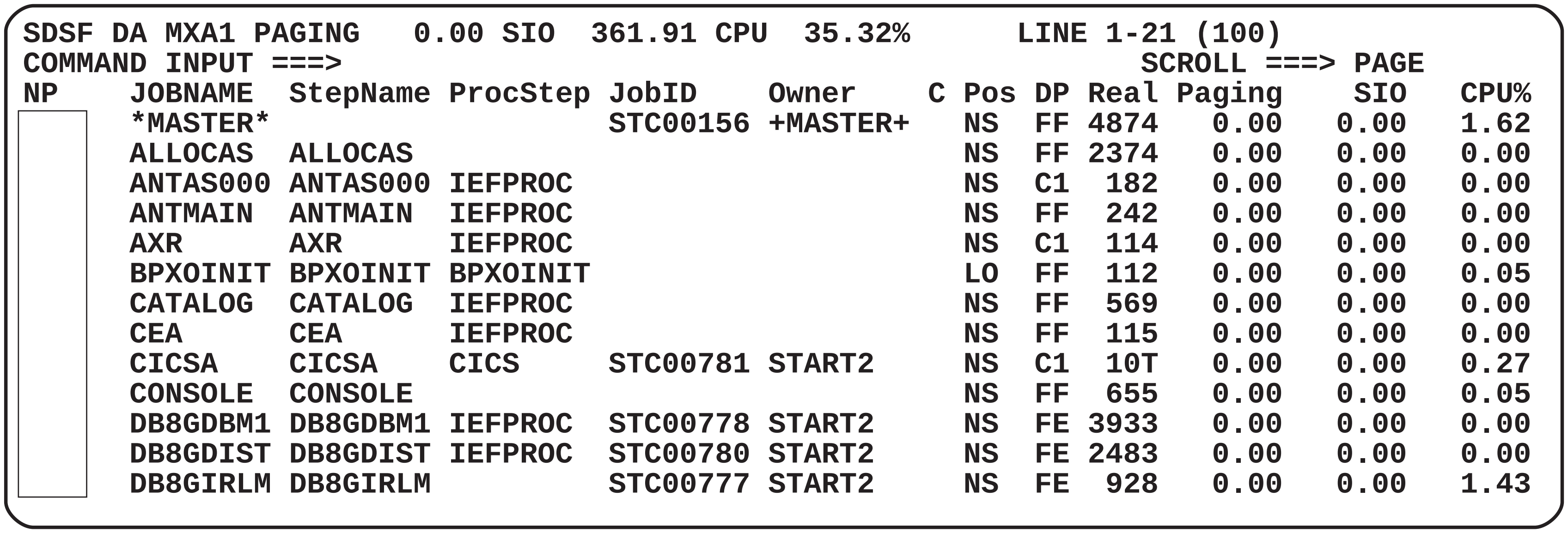
To change job class, put new class in the "C"' column (leave "NP" column blank)

SDSF - Display Active

- **Invoke this alone (===> DA) or with some options:**

===> DA OJOB	**(Only display batch jobs)**
===> DA OTSU	**(Only display TSO users)**
===> DA INIT	**(Include initiators on the display)**

- **Etc. (other options available, these are the most useful)**



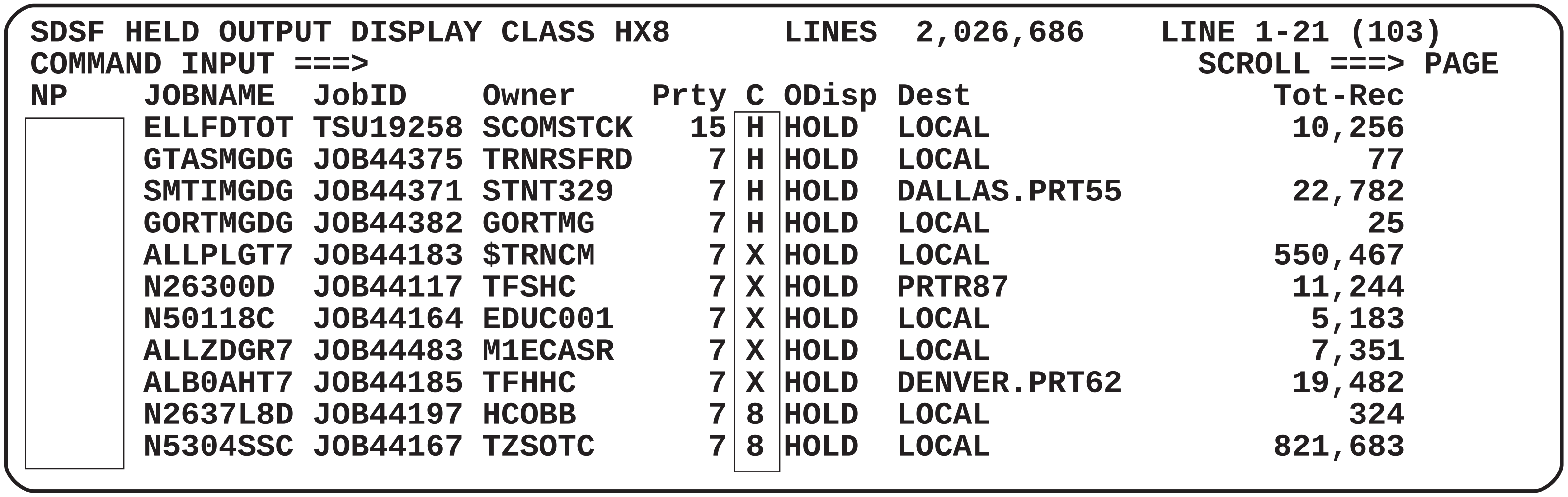
```
SDSF DA MXA1 PAGING   0.00 SIO  361.91 CPU  35.32%      LINE 1-21 (100)
COMMAND INPUT ===>                                             SCROLL ===> PAGE
NP     JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO    CPU%
       *MASTER*                   STC00156 +MASTER+   NS  FF 4874   0.00   0.00    1.62
       ALLOCAS  ALLOCAS                               NS  FF 2374   0.00   0.00    0.00
       ANTAS000 ANTAS000 IEFPROC                      NS  C1  182   0.00   0.00    0.00
       ANTMAIN  ANTMAIN  IEFPROC                      NS  FF  242   0.00   0.00    0.00
       AXR      AXR      IEFPROC                      NS  C1  114   0.00   0.00    0.00
       BPXOINIT BPXOINIT BPXOINIT                     LO  FF  112   0.00   0.00    0.05
       CATALOG  CATALOG  IEFPROC                      NS  FF  569   0.00   0.00    0.00
       CEA      CEA      IEFPROC                      NS  FF  115   0.00   0.00    0.00
       CICSA    CICSA    CICS     STC00781 START2     NS  C1  10T   0.00   0.00    0.27
       CONSOLE  CONSOLE                               NS  FF  655   0.00   0.00    0.05
       DB8GDBM1 DB8GDBM1 IEFPROC  STC00778 START2     NS  FE 3933   0.00   0.00    0.00
       DB8GDIST DB8GDIST IEFPROC  STC00780 START2     NS  FE 2483   0.00   0.00    0.00
       DB8GIRLM DB8GIRLM          STC00777 START2     NS  FE  928   0.00   0.00    1.43
```

"NP" action characters: S (select to browse), C (cancel), CD (cancel with a dump), P (cancel job and purge any SYSOUT data sets already created)

SDSF - Display TSO Held SYSOUT Data

❒ **Invoke this alone (===> H) or specify 1 to 7 SYSOUT classes:**

===> HHX8 **(Only display TSO SYSOUT hold classes H, X, and 8)**



```
SDSF HELD OUTPUT DISPLAY CLASS HX8      LINES  2,026,686    LINE 1-21 (103)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP    JOBNAME  JobID    Owner    Prty C ODisp Dest                Tot-Rec
      ELLFDTOT TSU19258 SCOMSTCK   15 H HOLD  LOCAL                10,256
      GTASMGDG JOB44375 TRNRSFRD    7 H HOLD  LOCAL                    77
      SMTIMGDG JOB44371 STNT329     7 H HOLD  DALLAS.PRT55         22,782
      GORTMGDG JOB44382 GORTMG      7 H HOLD  LOCAL                    25
      ALLPLGT7 JOB44183 $TRNCM      7 X HOLD  LOCAL               550,467
      N26300D  JOB44117 TFSHC       7 X HOLD  PRTR87               11,244
      N50118C  JOB44164 EDUC001     7 X HOLD  LOCAL                 5,183
      ALLZDGR7 JOB44483 M1ECASR     7 X HOLD  LOCAL                 7,351
      ALB0AHT7 JOB44185 TFHHC       7 X HOLD  DENVER.PRT62         19,482
      N2637L8D JOB44197 HCOBB       7 8 HOLD  LOCAL                   324
      N5304SSC JOB44167 TZSOTC      7 8 HOLD  LOCAL               821,683
```

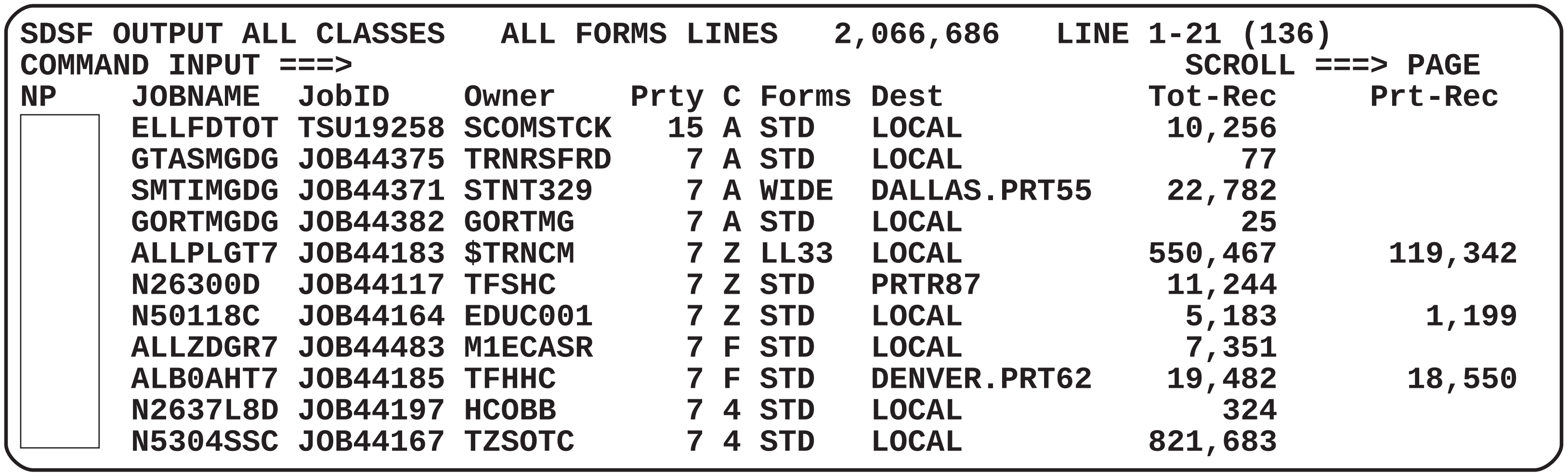
"NP" action characters: S (select to browse), P (purge the output), SB (browse using ISPF Browse), SE (use ISPF edit), SJ (select submited JCL using view), O (requeue the output to the class keyed in under the "OCLASS" column) (that is, to requeue, need "O" under "NP", and non-held SYSOUT class under "OCLASS")

Note that you may also overtype the PRTY, ODISP, DEST, and FORMS columns, if authorized

SDSF - Display Non-held SYSOUT Data

❒ **Invoke this alone (===> O) or specify 1 to 7 SYSOUT classes:**

===> OAZF47 **(Only display SYSOUT classes A, Z, F, 4, and 7)**



```
SDSF OUTPUT ALL CLASSES   ALL FORMS LINES   2,066,686   LINE 1-21 (136)
COMMAND INPUT ===>                                             SCROLL ===> PAGE
NP     JOBNAME  JobID    Owner    Prty C Forms Dest           Tot-Rec     Prt-Rec
       ELLFDTOT TSU19258 SCOMSTCK   15 A STD   LOCAL           10,256
       GTASMGDG JOB44375 TRNRSFRD    7 A STD   LOCAL               77
       SMTIMGDG JOB44371 STNT329     7 A WIDE  DALLAS.PRT55    22,782
       GORTMGDG JOB44382 GORTMG      7 A STD   LOCAL               25
       ALLPLGT7 JOB44183 $TRNCM      7 Z LL33  LOCAL          550,467      119,342
       N26300D  JOB44117 TFSHC       7 Z STD   PRTR87          11,244
       N50118C  JOB44164 EDUC001     7 Z STD   LOCAL            5,183        1,199
       ALLZDGR7 JOB44483 M1ECASR     7 F STD   LOCAL            7,351
       ALB0AHT7 JOB44185 TFHHC       7 F STD   DENVER.PRT62    19,482       18,550
       N2637L8D JOB44197 HCOBB       7 4 STD   LOCAL              324
       N5304SSC JOB44167 TZSOTC      7 4 STD   LOCAL          821,683
```

"NP" action characters: S (select to browse), SB (browse using ISPF Browse), SE (use ISPF edit), SJ (view submitted JCL), P (purge the output), H (hold the output from printing until explicitly released), A (release held output)

Other notes: Changing class in "C" column requeues the output to the indicated SYSOUT class (leave "NP" column blank); if authorized you may also typeover the PRTY, FORMS and DEST values, if you do so before the data set begins printing

Some Other SDSF Notes

- ❐ Notice you can examine the output from a job while it is running!
- ❐ Also, you can examine output from a job that is not TSO-held output, as long as the output hasn't been printed
- ❐ You can use the PREFIX command to restrict displays to jobnames beginning with one or more particular characters, or that follow a pattern:

```
PREFIX S00
PREFIX S00*
PREFIX I%8S*
```

  - ♦ To remove a prefix filter, issue PREFIX with no operands:

```
PREFIX
```

  - ♦ To query the current prefix, issue PREFIX with a '?'

```
PREFIX   ?
```

- ❐ The SET command can affect the displays, for example:

SET DISPLAY ON	- display status of PREFIX, OWNER, etc.
SET SCREEN	- Produces a fill-in-the-blanks panel to change screen colors, etc.

- ❐ The ARRANGE command lets you change the order of the columns on the display

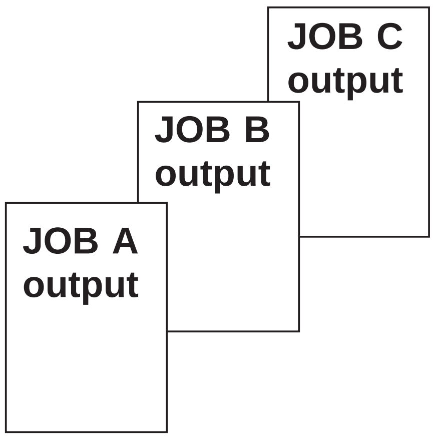
Some Other SDSF Notes, 2

❐ The OWNER command restricts displays to jobs with a particular TSO id, and you may query the setting, remove an OWNER filter, and use wildcards just as you can with PREFIX commands

```
OWNER S00
OWNER S00*
OWNER I%8S*
OWNER
OWNER   ?
```

❐ END (PF3) closes a displayed data set

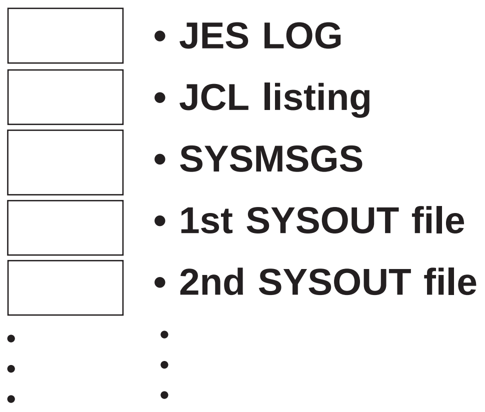
❐ If you select multiple jobs on the H or O queues, you will be shown the first one first; when you enter PF3 you will be shown the second one; and so on:



Some Other SDSF Notes, 3

❒ **While browsing output from a job, the command 'NEXT' moves you forward one whole SYSOUT file**

- ♦ **For example, JES2 Job Log to JCL listing, to Messages listing, to first SYSOUT file, *etc.*):**



❒ **NEXT *n* forward spaces you *n* SYSOUT files**

❒ **PREV [*n*] backs up *n* SYSOUT files (default is 1)**

- ♦ **NEXT may be abbreviated N, and PREV may be abbreviated P**

❒ **Placing a question mark (?) next to a jobname will produce a list of these individual files, and you can simply select (S) just the file(s) you want to see**

❒ **If you select a file or job using SE instead of S, you are put into View of the file and you may issue all View / Edit commands except SAVE!**

SDSF and the Action Bar

❒ **The SET SCREEN command allows you to display an action bar at the top of the screen (or to turn off the action bar)**

❒ **The Action Bar choices and their functions are:**

- **Display — request a particular panel; use instead of the SDSF primary commands H, I, ST, DA, etc.**

- **Filter — restrict rows displayed, based on values in one or more columns (can have multiple columns specified and use AND or OR connectives)**

  ✗ A pop-up window allows you specify these criteria

- **View — specify which columns to display on a panel, and in which order**

  ✗ May also be used to change the sort order of the rows on a display

- **Print — copy all or some sysout lines to another sysout file or a disk data set; print a screen image**

- **Options — set global SDSF options and settings**

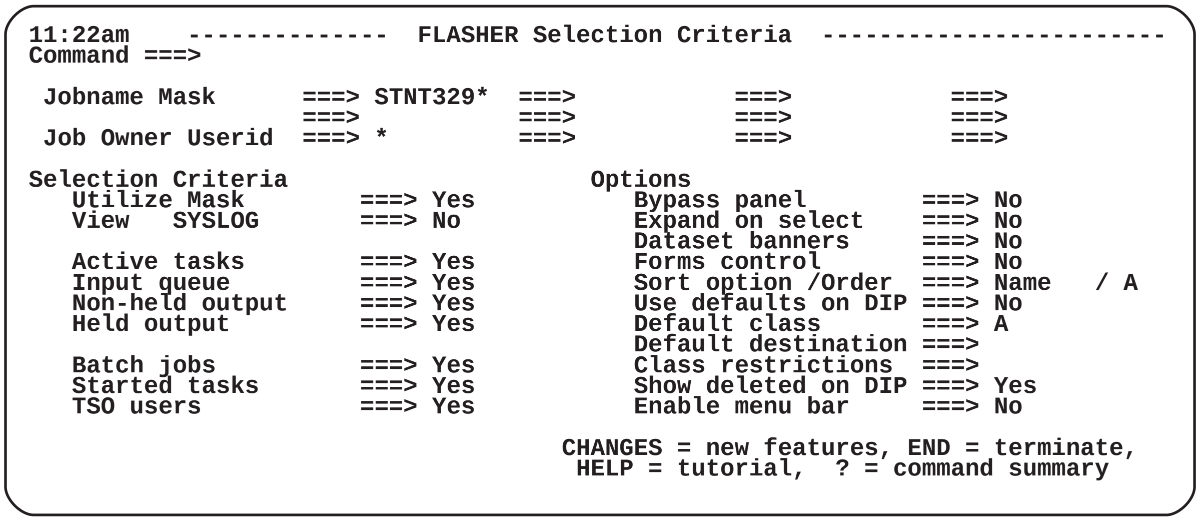
- **Help — access to SDSF help panels, SDSF tutorial, or BookManager**

❒ **This is a very rich product that deserves a lot of exploring**

This page intentionally left almost blank.

FLASHER

❒ You get to Flasher from any command / option line by coding the pseudo-command FLA, which brings you to the initial Flasher screen:



```
11:22am    -------------  FLASHER Selection Criteria  -----------------------
Command ===>

 Jobname Mask      ===> STNT329*  ===>           ===>           ===>
                   ===>           ===>           ===>           ===>
 Job Owner Userid  ===> *         ===>           ===>           ===>

Selection Criteria                     Options
   Utilize Mask        ===> Yes           Bypass panel        ===> No
   View   SYSLOG       ===> No            Expand on select    ===> No
                                          Dataset banners     ===> No
   Active tasks        ===> Yes           Forms control       ===> No
   Input queue         ===> Yes           Sort option /Order  ===> Name   / A
   Non-held output     ===> Yes           Use defaults on DIP ===> No
   Held output         ===> Yes           Default class       ===> A
                                          Default destination ===>
   Batch jobs          ===> Yes           Class restrictions  ===>
   Started tasks       ===> Yes           Show deleted on DIP ===> Yes
   TSO users           ===> Yes           Enable menu bar     ===> No

                                     CHANGES = new features, END = terminate,
                                      HELP = tutorial,  ? = command summary
```

❒ Use this screen to specify which jobs to display (by name pattern(s) or TSO id of submitter(s))

- Also identify which files you want to see from jobs that are displayed

- Can also specify if you want to see work in the input queues and other viewing and processing options

- Typically you set these parameters once then leave them

  ✗ Perhaps changing them briefly for special situations

- We examine a few of these options in more detail after you get more experience looking at jobs

FLASHER - Job List

❐ When you set your options on the initial Flasher panel and press Enter, you see the list of jobnames that match your selection criteria

- ♦ Here's a sample list from some recent work

```
01:07pm      CPU   Prt   Con   Init   Log -------------------  ROW 1 TO 6 OF 6
Command ===>                                                  Scroll ===> CSR
                      Flasher Jobname Selection  - V3R3M0
    Jobname  Number Date  Time  Owner    Class    Main     Pr Posn Hld  Status
    -------- ------ ----- ----- -------- -------- -------- -- ---- --- -------
    STNT329  T01093 0x/21 10:06 STNT329  DEFAULT  SYUB     15  411   0 Act Main
    STNT329G J14938 0x/21 13:05 STNT329  TSTMTEST           9  519   0 Avail MN
    STNT329M J14058 0x/21 11:38 STNT329  TSTMTEST           9  455   0 Wait WTR
    STNT329M J14258 0x/21 11:59 STNT329  DEFAULT            7   11   0 Wait WTR
    STNT329P J14071 0x/21 11:40 STNT329  TSTMTEST           9  456   0 Wait WTR
    STNT3293 J14405 0x/21 12:16 STNT329  TSTMTEST           9  471   0 Wait WTR
******************************* BOTTOM OF DATA ********************************
```

❐ As you press Enter, the screen is refreshed, particularly the message under the Status column (far right)

- ♦ Of course, it's better to wait for the Notify message before getting into Flasher

❐ When a job has completed, its status becomes Wait WTR - waiting for a Writer

- ♦ Although you can look at a job's output while it is running, we normally wait until a job is complete

❐ To look at a job's output, use line commands to the left of the jobname

- ♦ Although it doesn't show up easily, there are two one-position input fields to the left of the Jobname column

FLASHER Job List - Notes

❒ **The possible Status messages, and their meanings, are:**

Act CI	**Job is being processed by the Converter**
Act Main	**Job is running**
Avail MN	**Job is available to run**
Printing	**Non-held SYSOUT data is being printed**
Wait WTR	**Job has run; SYSOUT files are available to print**

❒ **Jobnames of the form T*nnnnn* represent the user's TSO session itself: that's you**

❒ **"Pr" represents print priority**

❒ **"Posn" represents relative place in the SYSOUT queues**

❒ **"Hld" indicates how many TSO data sets there are associated with a job**

FLASHER - Selecting Jobs

- **To examine job output, there are two major possibilities**
  - **Keying in a letter B or a letter S puts you in browse of the whole job's output**
    - The JES3 job log, JCL listing, system messages, and user SYSOUT reports look like a continuous string of lines, a single file
  - **Keying in a letter E expands the job's SYSOUT files into a list of the separate files**

- **Each approach has its place, and we'll be using both during the class**
  - **For our first job, simply Select the job to look at its entire output as a single file**

- **When you are in Browse of a job's output, you can use the standard Browse commands**
  - **Scrolling**
  - **Finding**
  - **END or RETURN to leave**

Deleting Job Outputs

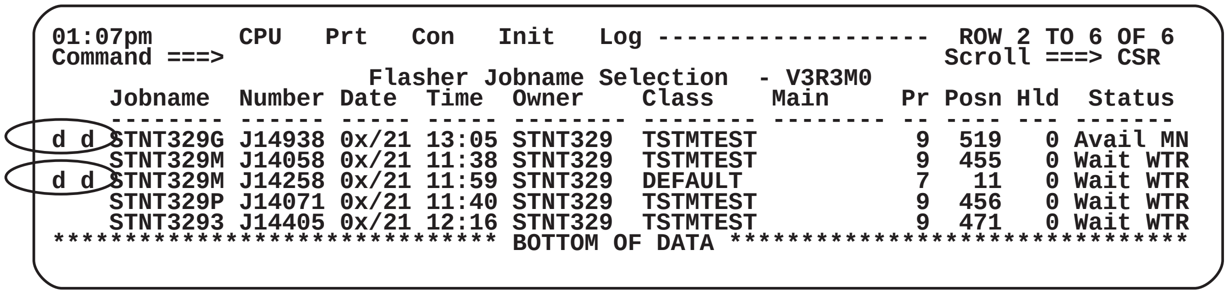
❒ After you have examined a job's output on Flasher, you probably want to either

- Delete the SYSOUT files

or

- Print the SYSOUT files

❒ To delete a job's output, from Flasher, enter two 'd's as shown below (it takes <u>two</u> 'd's so you are less likely to accidentally delete a job from the queues)



```
01:07pm      CPU   Prt   Con   Init   Log -------------------  ROW 2 TO 6 OF 6
Command ===>                                                  Scroll ===> CSR
                     Flasher Jobname Selection  - V3R3M0
    Jobname  Number Date  Time  Owner    Class    Main     Pr Posn Hld  Status
    -------- ------ ----- ----- -------- -------- -------- -- ---- --- -------
d d STNT329G J14938 0x/21 13:05 STNT329  TSTMTEST           9  519   0 Avail MN
    STNT329M J14058 0x/21 11:38 STNT329  TSTMTEST           9  455   0 Wait WTR
d d STNT329M J14258 0x/21 11:59 STNT329  DEFAULT            7   11   0 Wait WTR
    STNT329P J14071 0x/21 11:40 STNT329  TSTMTEST           9  456   0 Wait WTR
    STNT3293 J14405 0x/21 12:16 STNT329  TSTMTEST           9  471   0 Wait WTR
******************************* BOTTOM OF DATA ********************************
```

- Then press Enter

- Notice you can delete multiple jobs at a single time

Printing From Flasher

❒ **If you selected a job by 'e', for expand, you will have list of the JES3 files for your job, something like this:**

```
-------------------------  FLASHER Dataset Index  - V3R3M0--  ROW 1 TO 9 OF 9
Command ===>                                                  Scroll ===> CSR

    Jobname - STNT329M   Job Number - J12396             Show Deleted: Yes


    ### Ddname   Step     C-S Form     Dest     Wtr Name FCB  UCS  Copy Linecnt
    --- -------- -------- --- -------- -------- -------- ---- ---- ---- -------
    001 JESJCLIN          ?-N                            **** ****    1       0
    002 JESMSGLG          T-H STD1     ANYLOCAL          **** ****    1      50
    003 JESJCL            T-H STD1     ANYLOCAL          **** ****    1     107
    004 JESYSMSG          T-H STD1     ANYLOCAL          **** ****    1     169
    005 D0000009          ?-H                            **** ****    1       2
    006 SYSPRINT PROC0    T-H STD1     ANYLOCAL          **** ****    1    1767
    007 SYSPRINT PROC0    T-H STD1     ANYLOCAL          **** ****    1     414
    008 SYSPRINT PROC0    T-H STD1     ANYLOCAL          **** ****    1       3
    009 REPRT    PROC0    T-H                            **** ****    1       0
******************************* BOTTOM OF DATA ********************************
```

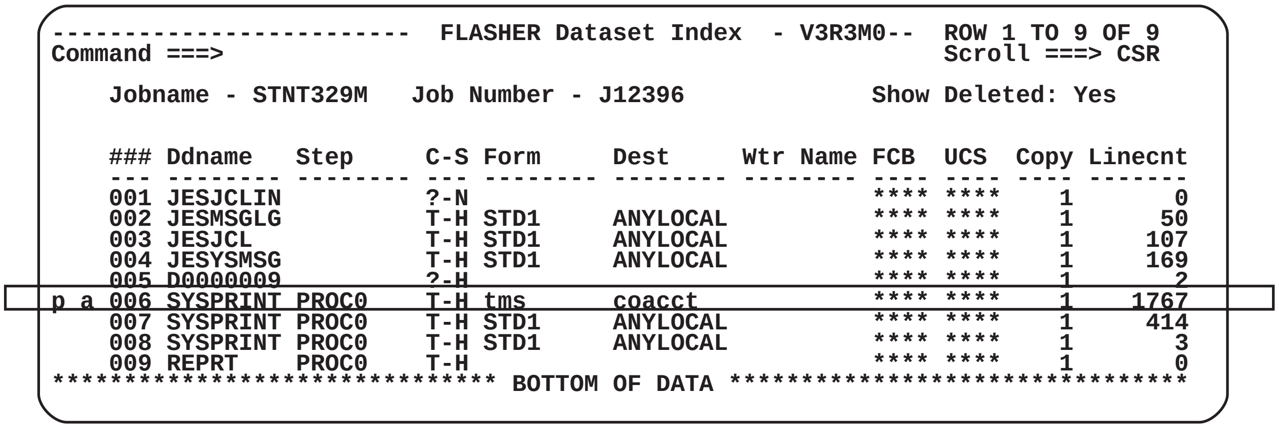
- **Generally speaking, you care about the SYSOUT files corresponding to DDnames you coded (the other DDnames are system-generated as part of the job running process)**

Printing From Flasher, continued

- **In the command area, next to the file(s) you want to print out, enter 'p' (for 'print') and the new SYSOUT class the file should have**

  ✗ You may also overtype the <u>Form</u> field (for some printers you must) and the <u>Dest</u> field

- **For example:**



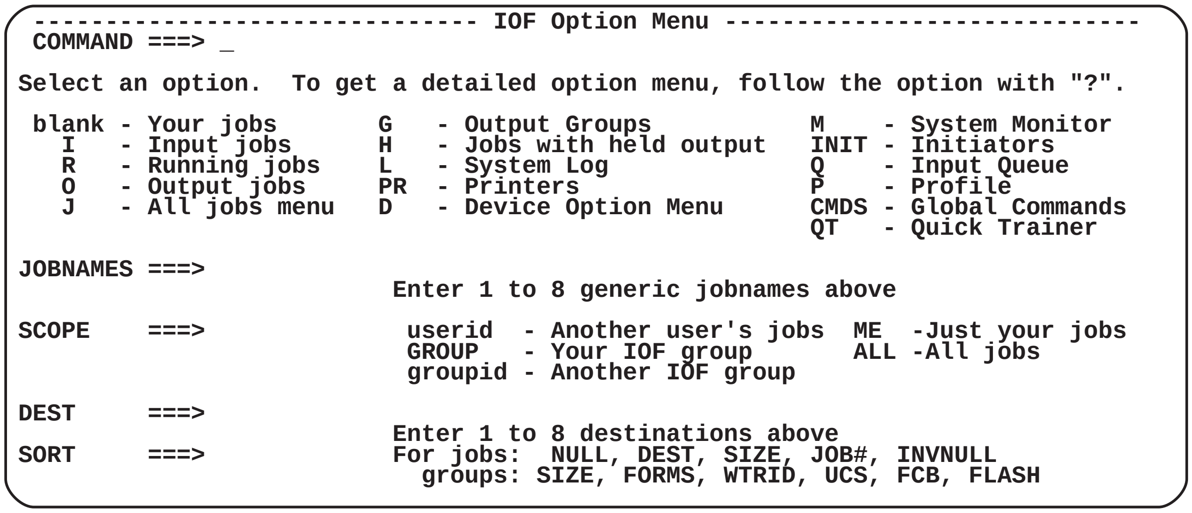
```
-------------------------  FLASHER Dataset Index  - V3R3M0--  ROW 1 TO 9 OF 9
Command ===>                                                 Scroll ===> CSR

    Jobname - STNT329M   Job Number - J12396             Show Deleted: Yes


    ### Ddname   Step     C-S Form     Dest     Wtr Name FCB  UCS  Copy Linecnt
    --- -------- -------- --- -------- -------- -------- ---- ---- ---- -------
    001 JESJCLIN          ?-N                            **** ****    1       0
    002 JESMSGLG          T-H STD1     ANYLOCAL          **** ****    1      50
    003 JESJCL            T-H STD1     ANYLOCAL          **** ****    1     107
    004 JESYSMSG          T-H STD1     ANYLOCAL          **** ****    1     169
    005 D0000009          ?-H                            **** ****    1       2
p a 006 SYSPRINT PROC0    T-H tms      coacct            **** ****    1    1767
    007 SYSPRINT PROC0    T-H STD1     ANYLOCAL          **** ****    1     414
    008 SYSPRINT PROC0    T-H STD1     ANYLOCAL          **** ****    1       3
    009 REPRT    PROC0    T-H                            **** ****    1       0
******************************* BOTTOM OF DATA ********************************
```

- **You may also print or requeue from the Flasher Job List**

- **In any case, if you don't specify new values (output class, Form, Dest, and so on), defaults are taken from the Flasher entry panel**

IOF

❐ **You get to IOF from any command / option line by entering IOF, which brings you to the initial IOF screen:**



```
 ------------------------------ IOF Option Menu -----------------------------
 COMMAND ===> _

Select an option.  To get a detailed option menu, follow the option with "?".

 blank - Your jobs       G   - Output Groups          M    - System Monitor
   I   - Input jobs      H   - Jobs with held output  INIT - Initiators
   R   - Running jobs    L   - System Log             Q    - Input Queue
   O   - Output jobs     PR  - Printers               P    - Profile
   J   - All jobs menu   D   - Device Option Menu     CMDS - Global Commands
                                                      QT   - Quick Trainer

JOBNAMES ===>
                          Enter 1 to 8 generic jobnames above

SCOPE    ===>              userid  - Another user's jobs  ME  -Just your jobs
                           GROUP   - Your IOF group       ALL -All jobs
                           groupid - Another IOF group

DEST     ===>
                          Enter 1 to 8 destinations above
SORT     ===>             For jobs:  NULL, DEST, SIZE, JOB#, INVNULL
                            groups: SIZE, FORMS, WTRID, UCS, FCB, FLASH
```

♦ **If you press <Enter> without keying in anything else, you will go to the IOF Job List Menu with a list of all jobs whose names begin with your TSO ID**

✗ To see other jobs, fill in JOBNAMES or SCOPE (or both) to specify what jobs you want to see

✗ For SORT, you can specify the sort order of the job list

❐ **Note that if you select option P, you are shown a list of ways you can tailor an IOF profile just for you**

♦ **One of these options, 1, lets you set your default print destination, your default SYSOUT release class, as well as a few other less interesting parameters**

❐ **Note also that this panel may look different, depending on options chosen by your installation**

IOF, continued

❒ **At some point you get a list of jobs for you to work with, something like this:**

```
------------------------------ IOF Job List Menu ------------( 9 )--------------
COMMAND ===> _                                                SCROLL ===> CURSOR
--------------------------------- Input Jobs -----------------------------------
-------JOBNAME--JOBID---ACT-STAT-OWNER----C-POSIT-PRTY-SRVCLASS-QUALIFIER-------
_    1 DOUBLE8  J00312      JOB  SCOMSTO  A   2     9
-------------------------------- Running Jobs ----------------------------------
-------JOBNAME--JOBID---ACT-STAT-SYID-------CPU-------I/O-STEP------PROCSTEP-SWP
_    2 DOUBLEA J00308            THOR
_    3 DOUBLEY J00309            SYB      12:23    256.05 ARRGH     BUTTER   IN
--------------------------------- Output Jobs ----------------------------------
-------JOBNAME--JOBID---ACT-STAT-OWNER----DEST/DEVICE--------RECS-HELD-DAY--TIME
_    4 TROUBLEA J00305           TROUBLE  LOCAL               300      120 08:32
_    5 TROUBLEA J00333           TROUBLE  LOCAL               12K   2K 120 08:50
_    6 TROUBLEA J00800           TROUBLE  LOCAL               310      121 07:13
_    7 TROUBLED J00889           TROUBLE  LOCAL             1034K      120 08:12
_    8 TROUBLEK J01012           TROUBLE  LOCAL                5K      120 10:15
_    9 TROUBLES J01015           TROUBLE  LOCAL               220      120 10:44
_   10 TROUBLEX J01020           TROUBLE  LOCAL               13K   6K 120 11:27
```

- **Display output from a job in one of these ways**

  ✗ On the Command line, type in the list number (1-10 in the example) of the job you are interested in and press <Enter>

  ✗ On the line command field next to the job you want to display, type in a letter "s" and press <Enter>

  ➢ This lists all your SYSOUT-type data sets as separate files you can work with (see next page)

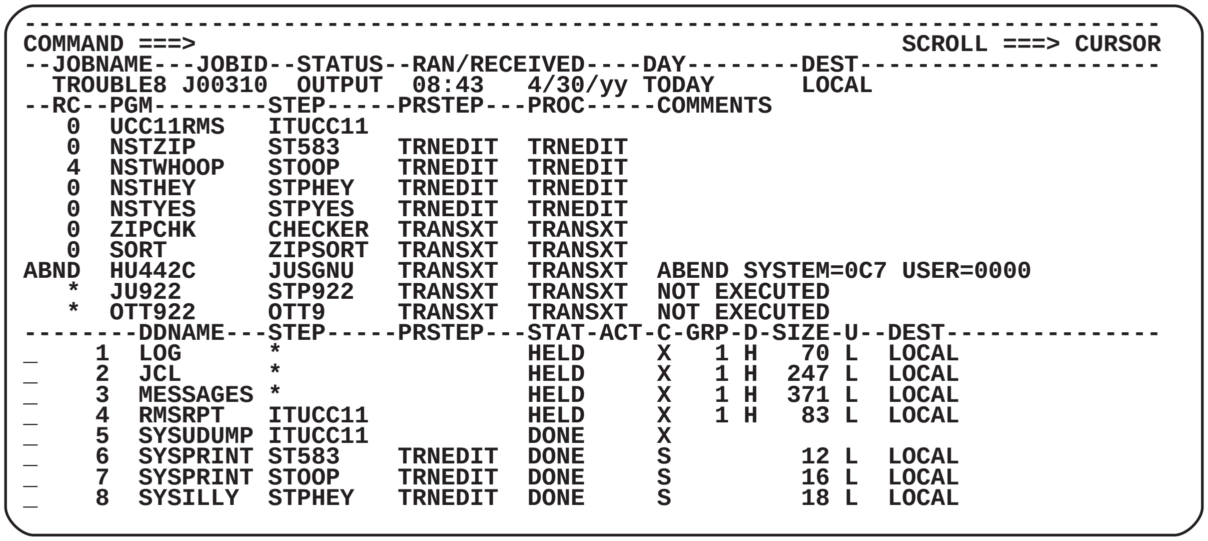
  ✗ On the line command field next to the job you want to display, type in a letter "b" and press <Enter>

  ➢ This puts you in Browse of all the SYSOUT-type files from the job as a single file

IOF, continued

❒ **If you display a job using "b" option, you are in standard Browse of the entire job output**

❒ **If you display a job using "s", or by selecting the job's list number in the Command line, you get a list of the steps in the job and all the SYSOUT files produced; for example:**



```
--------------------------------------------------------------------------------
COMMAND ===>                                                 SCROLL ===> CURSOR
--JOBNAME---JOBID--STATUS--RAN/RECEIVED----DAY--------DEST---------------------
  TROUBLE8 J00310  OUTPUT  08:43   4/30/yy TODAY      LOCAL
--RC--PGM--------STEP-----PRSTEP---PROC-----COMMENTS
   0  UCC11RMS   ITUCC11
   0  NSTZIP     ST583    TRNEDIT  TRNEDIT
   4  NSTWHOOP   STOOP    TRNEDIT  TRNEDIT
   0  NSTHEY     STPHEY   TRNEDIT  TRNEDIT
   0  NSTYES     STPYES   TRNEDIT  TRNEDIT
   0  ZIPCHK     CHECKER  TRANSXT  TRANSXT
   0  SORT       ZIPSORT  TRANSXT  TRANSXT
ABND  HU442C     JUSGNU   TRANSXT  TRANSXT  ABEND SYSTEM=0C7 USER=0000
   *  JU922      STP922   TRANSXT  TRANSXT  NOT EXECUTED
   *  OTT922     OTT9     TRANSXT  TRANSXT  NOT EXECUTED
--------DDNAME---STEP-----PRSTEP---STAT-ACT-C-GRP-D-SIZE-U--DEST---------------
_    1  LOG      *                 HELD     X   1 H   70 L  LOCAL
_    2  JCL      *                 HELD     X   1 H  247 L  LOCAL
_    3  MESSAGES *                 HELD     X   1 H  371 L  LOCAL
_    4  RMSRPT   ITUCC11           HELD     X   1 H   83 L  LOCAL
_    5  SYSUDUMP ITUCC11           DONE     X
_    6  SYSPRINT ST583    TRNEDIT  DONE     S         12 L  LOCAL
_    7  SYSPRINT STOOP    TRNEDIT  DONE     S         16 L  LOCAL
_    8  SYSILLY  STPHEY   TRNEDIT  DONE     S         18 L  LOCAL
```

Notes

♦ **Under the RC column is the return code for every step**

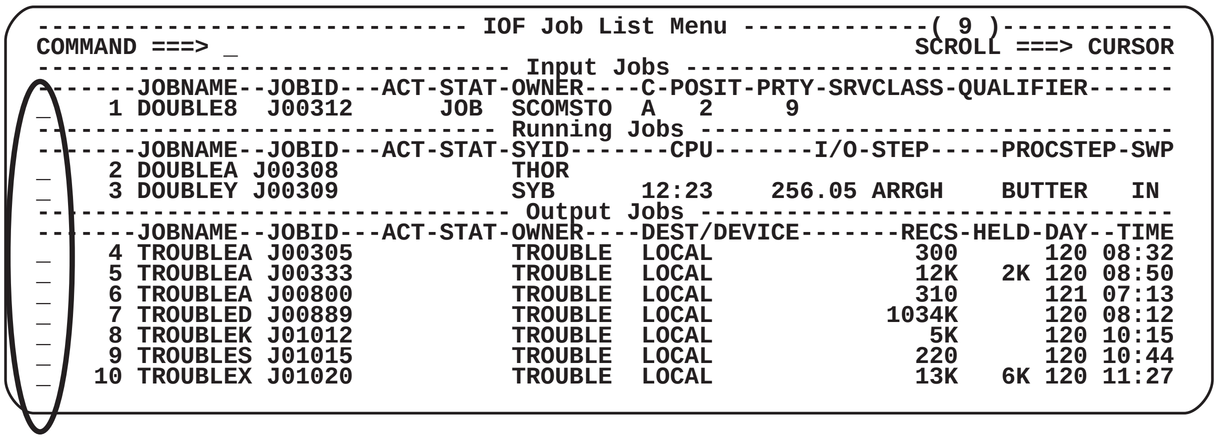
✗ Generally, small return codes are good ("0" traditionally implies no errors, and "4" warnings, etc.)

✗ A return code of ABND is not good: it says that program "blew up"; normally, all steps after an abend are flushed

♦ **Examine individual data sets by coding the data set list number in the Command line, or keying an "s" next to the DDname and pressing <Enter>**

IOF Commands

- When you are done looking at a job's output, from the Job List Menu you can dispose of the job by keying in a line command next to the job name



```
------------------------------ IOF Job List Menu -------------( 9 )------------
COMMAND ===> _                                               SCROLL ===> CURSOR
--------------------------------- Input Jobs ----------------------------------
-------JOBNAME--JOBID---ACT-STAT-OWNER----C-POSIT-PRTY-SRVCLASS-QUALIFIER------
_    1 DOUBLE8  J00312      JOB  SCOMSTO  A   2     9
-------------------------------- Running Jobs ---------------------------------
-------JOBNAME--JOBID---ACT-STAT-SYID-------CPU-------I/O-STEP-----PROCSTEP-SWP
_    2 DOUBLEA J00308            THOR
_    3 DOUBLEY J00309            SYB      12:23    256.05 ARRGH    BUTTER   IN
--------------------------------- Output Jobs ---------------------------------
-------JOBNAME--JOBID---ACT-STAT-OWNER----DEST/DEVICE-------RECS-HELD-DAY--TIME
_    4 TROUBLEA J00305            TROUBLE  LOCAL              300       120 08:32
_    5 TROUBLEA J00333            TROUBLE  LOCAL              12K   2K 120 08:50
_    6 TROUBLEA J00800            TROUBLE  LOCAL              310       121 07:13
_    7 TROUBLED J00889            TROUBLE  LOCAL            1034K       120 08:12
_    8 TROUBLEK J01012            TROUBLE  LOCAL               5K       120 10:15
_    9 TROUBLES J01015            TROUBLE  LOCAL              220       120 10:44
_   10 TROUBLEX J01020            TROUBLE  LOCAL              13K   6K 120 11:27
```

❒ **Most useful commands:**

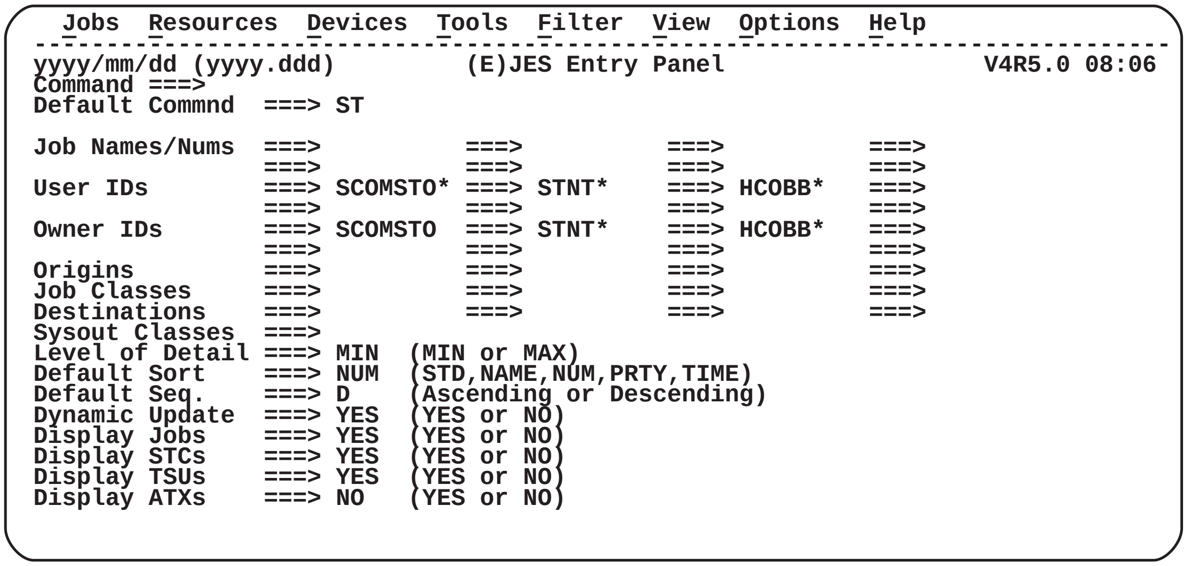
- S and B - already discussed
- C - cancel a job and purge its ouput
- PR - print the job; can overtype the DEST if necessary
- R - requeue the job; job is routed to the default SYSOUT class and destination in your IOF Profile
- H - hold a job; do not let it start running
- A - release a held job

❒ **You can also issue these commands on the command line, for example:**

COMMAND ===> 1-5c	cancel jobs 1, 2, 3, 4, 5
COMMAND ===> 1,5c	cancel jobs 1, 5
COMMAND ===> 1,5-7c	cancel jobs 1, 5, 6, 7

(E)JES

❒ **You get to (E)JES from any command / option line by coding an installation specific option or command (by default: EJ), which brings you to the initial (E)JES screen:**



```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
----------------------------------------------------------------------------
yyyy/mm/dd (yyyy.ddd)         (E)JES Entry Panel                V4R5.0 08:06
Command ===>
Default Commnd  ===> ST

Job Names/Nums  ===>          ===>          ===>          ===>
                ===>          ===>          ===>          ===>
User IDs        ===> SCOMSTO* ===> STNT*    ===> HCOBB*   ===>
                ===>          ===>          ===>          ===>
Owner IDs       ===> SCOMSTO  ===> STNT*    ===> HCOBB*   ===>
                ===>          ===>          ===>          ===>
Origins         ===>          ===>          ===>          ===>
Job Classes     ===>          ===>          ===>          ===>
Destinations    ===>          ===>          ===>          ===>
Sysout Classes  ===>
Level of Detail ===> MIN  (MIN or MAX)
Default Sort    ===> NUM  (STD,NAME,NUM,PRTY,TIME)
Default Seq.    ===> D    (Ascending or Descending)
Dynamic Update  ===> YES  (YES or NO)
Display Jobs    ===> YES  (YES or NO)
Display STCs    ===> YES  (YES or NO)
Display TSUs    ===> YES  (YES or NO)
Display ATXs    ===> NO   (YES or NO)
```

❒ **Use this screen to specify which jobs to display (by name pattern(s) or TSO id of submitter(s))**

- ♦ **Also identify which files you want to see from jobs that are displayed**

- ♦ **Can also specify if you want to see work in the input queues and other viewing and processing options**

- ♦ **Typically you set these parameters once then leave them**

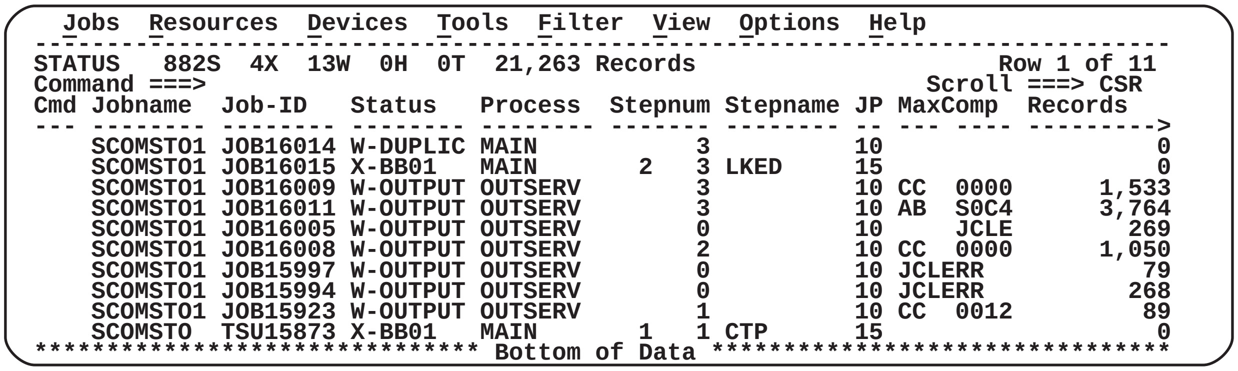
  - ✗ Perhaps changing them briefly for special situations

- ♦ **We'll examine a few of these options in more detail after you get more experience looking at jobs**

(E)JES - Job List

❒ When you set your options on the initial (E)JES panel and press Enter, you see the list of jobnames that match your selection criteria

♦ Here's a sample list from some recent work



```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
-------------------------------------------------------------------------------
STATUS   882S  4X  13W  0H  0T  21,263 Records                     Row 1 of 11
Command ===>                                                  Scroll ===> CSR
Cmd Jobname  Job-ID   Status   Process  Stepnum Stepname JP MaxComp  Records
--- -------- -------- -------- -------- ------- -------- -- --- ---- --------->
    SCOMSTO1 JOB16014 W-DUPLIC MAIN           3          10                   0
    SCOMSTO1 JOB16015 X-BB01   MAIN       2   3 LKED     15                   0
    SCOMSTO1 JOB16009 W-OUTPUT OUTSERV        3          10 CC  0000      1,533
    SCOMSTO1 JOB16011 W-OUTPUT OUTSERV        3          10 AB  S0C4      3,764
    SCOMSTO1 JOB16005 W-OUTPUT OUTSERV        0          10     JCLE        269
    SCOMSTO1 JOB16008 W-OUTPUT OUTSERV        2          10 CC  0000      1,050
    SCOMSTO1 JOB15997 W-OUTPUT OUTSERV        0          10 JCLERR           79
    SCOMSTO1 JOB15994 W-OUTPUT OUTSERV        0          10 JCLERR          268
    SCOMSTO1 JOB15923 W-OUTPUT OUTSERV        1          10 CC  0012         89
    SCOMSTO  TSU15873 X-BB01   MAIN       1   1 CTP      15                   0
******************************* Bottom of Data ********************************
```

❒ As you press Enter, the screen is refreshed, particularly the message under the Status column

♦ Of course, it's better to wait for the Notify message before getting into (E)JES

❒ When a job has completed, its status becomes W-OUTPUT waiting for a Writer

♦ Although you can look at a job's output while it is running, we normally wait until a job is complete

❒ To look at job output, use line commands to the left of the jobname

❒ Note that (E)JES was originally only for JES3, but now it supports JES2 also; appearances differ somewhat on these two platforms

(E)JES Job List - Notes

❒ **Some possible Status messages, and their meanings, are:**

X-*node*	**Job is running (executing) at** *node*
X-CI *ss*	**Job is being converted by subsystem** *ss*
H-OPER	**Job has been held by the operator**
OUTSERV	**Non-held SYSOUT data is being printed**
W-OUTPUT	**Job has run; SYSOUT files are available to print or view**
W-DUPLIC	**Job is held because there is another job with the same jobname currently running**

❒ **Job numbers of the form TSU***nnnnn* **represent the user's TSO session itself: that's you**

❒ **"Jp" represents job's priority**

(E)JES - Selecting Jobs

❒ **To examine job output, there are two major possibilities**

- ♦ **Keying in a letter B puts you in browse of the whole job's output, V lets you view the output**

  - ✗ The JES3 job log, JCL listing, system messages, and user SYSOUT reports look like a continuous string of lines, a single file

- ♦ **Keying in a letter S expands the job SYSOUT files into a list of the separate files**

❒ **Each approach has its place, and we'll be using both during the class**

- ♦ **For our first job, simply Browse the job to look at its entire output as a single file**

❒ **When you are in View or Browse of a job's output, you can use the standard View or Browse commands**

- ♦ **Scrolling**

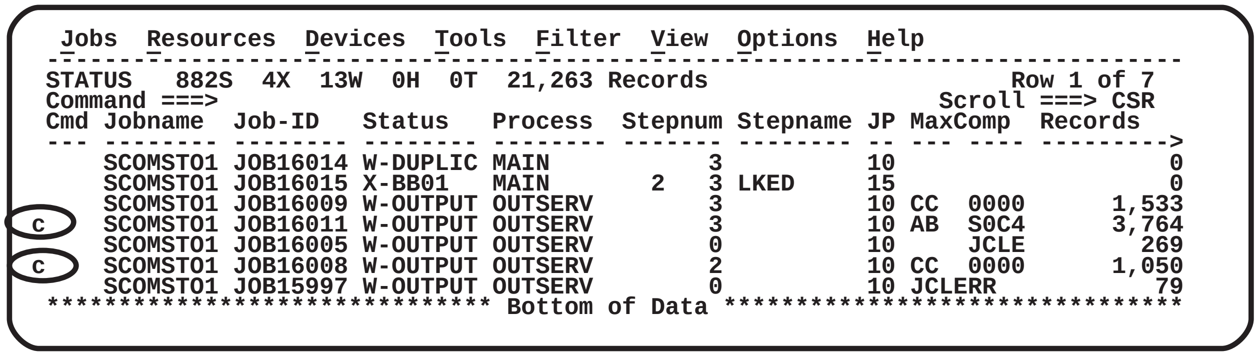
- ♦ **Finding**

- ♦ **END or RETURN to leave**

Deleting Job Outputs

❐ **After you have examined a job's output on (E)JES, you probably want to either delete the SYSOUT files or print the SYSOUT files**

❐ **To delete a job's output, from (E)JES, enter "c", for Cancel:**

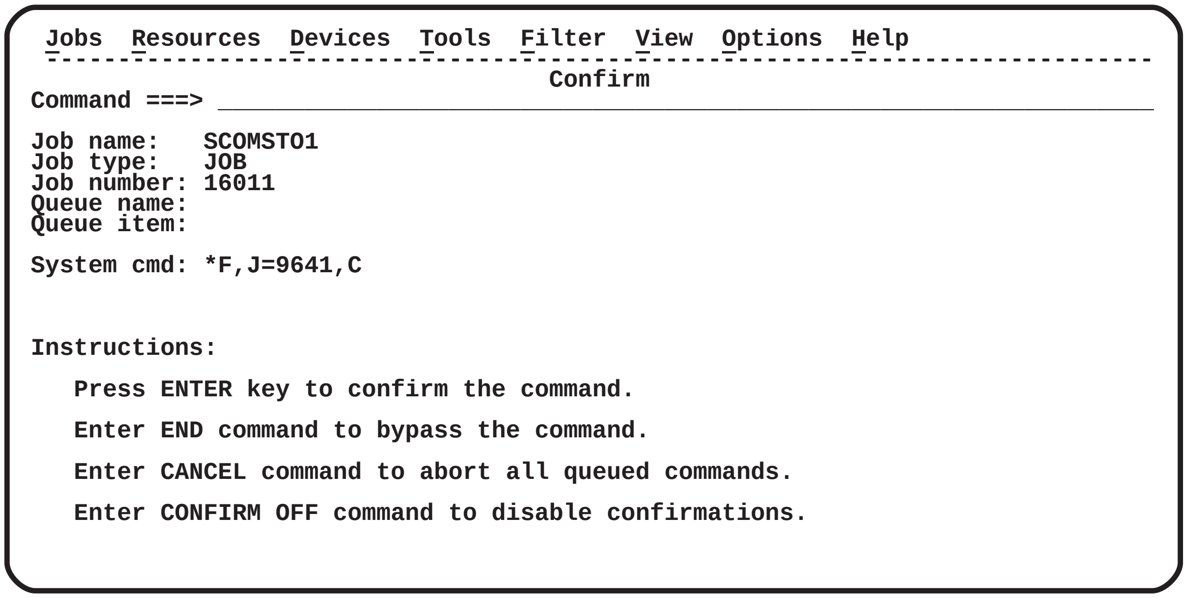


```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
 -------------------------------------------------------------------------------
 STATUS   882S  4X  13W  0H  0T  21,263 Records                    Row 1 of 7
 Command ===>                                                  Scroll ===> CSR
 Cmd Jobname  Job-ID   Status   Process  Stepnum Stepname JP MaxComp  Records
 --- -------- -------- -------- -------- ------- -------- -- --- ---- --------->
     SCOMSTO1 JOB16014 W-DUPLIC MAIN           3          10                  0
     SCOMSTO1 JOB16015 X-BB01   MAIN       2   3 LKED     15                  0
     SCOMSTO1 JOB16009 W-OUTPUT OUTSERV        3          10 CC  0000     1,533
  c  SCOMSTO1 JOB16011 W-OUTPUT OUTSERV        3          10 AB  S0C4     3,764
     SCOMSTO1 JOB16005 W-OUTPUT OUTSERV        0          10     JCLE       269
  c  SCOMSTO1 JOB16008 W-OUTPUT OUTSERV        2          10 CC  0000     1,050
     SCOMSTO1 JOB15997 W-OUTPUT OUTSERV        0          10 JCLERR          79
 ******************************* Bottom of Data ********************************
```

♦ **Notice you can delete multiple jobs at a single time**

♦ **Then press Enter**

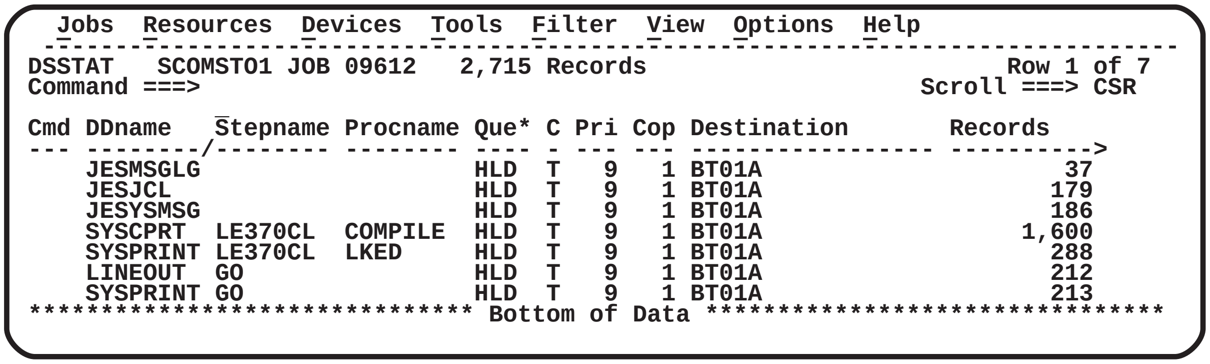
♦ **You will see a panel asking you to confirm the cancel:**



```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
 -----------------------------------------------------------------------------
                                    Confirm
 Command ===> ________________________________________________________________

 Job name:   SCOMSTO1
 Job type:   JOB
 Job number: 16011
 Queue name:
 Queue item:

 System cmd: *F,J=9641,C


 Instructions:

    Press ENTER key to confirm the command.

    Enter END command to bypass the command.

    Enter CANCEL command to abort all queued commands.

    Enter CONFIRM OFF command to disable confirmations.
```

Printing From (E)JES

❒ **If you selected a job by "s", you will have list of the JES files for your job, something like this:**



```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
 -------------------------------------------------------------------------------
DSSTAT   SCOMST01 JOB 09612   2,715 Records                         Row 1 of 7
Command ===>                                                    Scroll ===> CSR

Cmd DDname   Stepname Procname Que* C Pri Cop Destination       Records
--- --------/-------- -------- ---- - --- --- ---------------- --------->
    JESMSGLG                   HLD  T   9   1 BT01A                   37
    JESJCL                     HLD  T   9   1 BT01A                  179
    JESYSMSG                   HLD  T   9   1 BT01A                  186
    SYSCPRT  LE370CL  COMPILE  HLD  T   9   1 BT01A                1,600
    SYSPRINT LE370CL  LKED     HLD  T   9   1 BT01A                  288
    LINEOUT  GO                HLD  T   9   1 BT01A                  212
    SYSPRINT GO                HLD  T   9   1 BT01A                  213
******************************* Bottom of Data ********************************
```

- **Generally speaking, you care about the SYSOUT files corresponding to DDnames you coded (the other DDnames are system-generated as part of the job running process)**

❒ **Now, with (E)JES, you can either requeue a SYSOUT file to a different output queue**

- **Key in w*c* (where *c* is the SYSOUT class to use) next to the file(s) you want to requeue ('w' for 'write')**

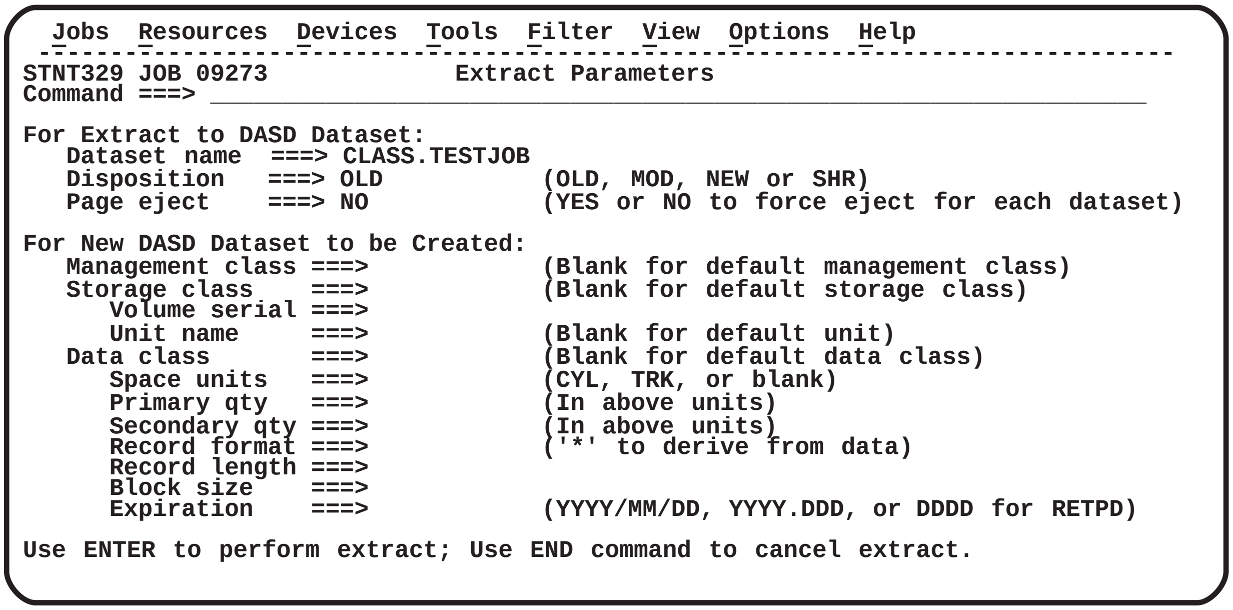
❒ **Or you issue the Extract command**

- **Key in e*x* (where *x* is "d" for a DASD data set, "1" for the first print data set, "2" for the second print data set), next to the file(s) you want to print [you can define one or two sets of parameters for print datasets, here you identify which one to use]**

  ✗ This will give you another panel to fill in information ...

Printing From (E)JES, continued

❒ **If you asked to extract to a DASD data set, you see a screen like this:**



```
  Jobs  Resources  Devices  Tools  Filter  View  Options  Help
 -----------------------------------------------------------------------------
STNT329 JOB 09273             Extract Parameters
Command ===> _________________________________________________________________

For Extract to DASD Dataset:
   Dataset name  ===> CLASS.TESTJOB
   Disposition   ===> OLD           (OLD, MOD, NEW or SHR)
   Page eject    ===> NO            (YES or NO to force eject for each dataset)

For New DASD Dataset to be Created:
   Management class ===>            (Blank for default management class)
   Storage class    ===>            (Blank for default storage class)
      Volume serial ===>
      Unit name     ===>            (Blank for default unit)
   Data class       ===>            (Blank for default data class)
      Space units   ===>            (CYL, TRK, or blank)
      Primary qty   ===>            (In above units)
      Secondary qty ===>            (In above units)
      Record format ===>            ('*' to derive from data)
      Record length ===>
      Block size    ===>
      Expiration    ===>            (YYYY/MM/DD, YYYY.DDD, or DDDD for RETPD)

Use ENTER to perform extract; Use END command to cancel extract.
```

❒ **If you asked to spin off to a print data set, you see a similar screen for you to enter destination and other print characteristics**

❒ **In either, case, the SYSOUT data also remains on the SPOOL**

❒ **You may also print or requeue from the (E)JES Job List, to get an entire job's output printed or saved to disk**

Parameters Related To Listing Routing

- ❒ **Instead of changing parameters through SDSF, IOF, Flasher, or (E)JES, you may specify them directly on the DD statement for a SYSOUT file**

  - ♦ **In this way the listing(s) are automatically sent to the location of your choice**

- ❒ **The possible routing parameters are:**

  SYSOUT=*x*

  - ♦ **Specify the output class the report is to go to automatically**

  - ♦ **Choosing a non-TSO-Held SYSOUT class routes a file to a place where a Writer program can eventually write it to hardcopy**

  DEST=value

  - ♦ **Specify one of the following** value**s**

    - ✗ ANYLOCAL - any local device attached to the global processor

    - ✗ name - any installation defined name that identifies a pool of one or more devices

    - ✗ (node,TSO_id) - an installation specified network node name and a TSO id valid at that node; file is transferred to that user's SPOOL file

Computer Exercise: Utilities

Code and execute the JCL and control statements necessary to perform the following functions:

Step One

Use IEBGENER to create a copy of INPUTA under your userid (call the new data set "userid.TRAIN.INPUTA"). Two tracks of space is plenty.

Step Two

Use IDCAMS to:

print records 20 through 50 from your copy of INPUTA in character format. (PRINT command)

rename "userid.TRAIN.INPUTA" to "userid.TRAIN.INPUTB" (ALTER)

then rename it back. (ALTER)

Step Three

Use IEFBR14 to delete your copy of INPUTA.

Exercise Stretch: In Step Three, add a DD statement to allocate "userid.TRAIN.INPUTZ" like INPUTA; then add a Step Four, use IDCAMS to copy **my copy** of INPUTA (______________TRAIN.INPUTA) into "userid.TRAIN.INPUTZ", then print records 120 through 150 of INPUTZ, then delete INPUTZ.

Section Preview

❒ SORT Program

- SORT Capabilities
- JCL for SORT
- SORT control statements
- Using SORT to do a copy
- Using the SORT (Machine Exercise)

SORT Capabilities

❐ **Sort one file into a new sequence (SORT statement)**

❐ **Merge several files (already in the same relative sequence) into one file (MERGE statement)**

❐ **Copy a file (SORT, MERGE, or OPTION statement)**

❐ **Inputs: Sequential, member of PDS or PDSE, VSAM, or HFS files**

❐ **Outputs: Sequential, member of PDS or PDSE, VSAM, or HFS**

- ♦ **VSAM files must be pre-defined unless SMS is installed and active in your system**

❐ **Sort on subsets**

- ♦ **Only records that meet some criteria (INCLUDE statement)**
- ♦ **All records except those that meet some criteria (OMIT statement)**

❐ **Reformat records**

- ♦ **Before sorting (INREC statement)**
- ♦ **After sorting (OUTREC statement)**

SORT Control

JCL

```
//SORTIT     EXEC    PGM=SORT
//SORTIN     DD      point to the file to be sorted        (may be concatenated inputs)
//SORTOUT    DD      point to the output location
//SYSIN      DD      for Sort control statements
//SYSOUT     DD      SYSOUT=*                               for Sort messages
.
.                                                           There are additional DD
.                                                           statements possible; not
                                                            discussed here
```

Sort control statements

INCLUDE	code content in columns 2-71, inclusive
OMIT	
INREC	indicate continuation by comma at end of
SORT	last operand on a line;
OUTREC	continuation lines coded in 2-71 also
	others, not discussed here

Introduction to INCLUDE / OMIT Statements

❐ **INCLUDE: only sort, copy, or merge records that meet the condition test(s)**

❐ **OMIT: sort, copy, or merge all records except those that meet the condition test(s)**

- ♦ **May only specify one INCLUDE or one OMIT; may not specify one of each**

❐ **Condition test(s) specified as "COND=" and a series of one or more tests:**

Input record <u>field</u>, <u>Relationship</u>, Input record <u>field</u>

or

Input record <u>field</u>, <u>Relationship</u>, <u>Constant</u>

<u>Where</u>

- ♦ **<u>field</u> is specified as** *starting_location*, *length*, **and** *data_type* **(CH, PD, ZD, FI, BI,** *etc.***)**
- ♦ **<u>Relationship</u> is one of: EQ (equal), NE (not equal), GT (greater than), GE (greater than or equal), LT (less than), or LE (less than or equal to)**
- ♦ **<u>Constant</u> is a numeric literal, a character string (C'...'), or a hex string (X'...')**

❐ **Tests are separated by "AND", or "&" or "OR" or "|"**

Introduction to INCLUDE / OMIT Statements, Examples

```
OMIT   COND=(34,5,CH,NE,C'ENTRY',AND,6,4,FI,GE,20,4,FI,
                                          OR,1,1,CH, EQ,X'FF')
```

Note that ANDs are evaluated before ORs; you can use parentheses to direct the order of evaluation

```
INCLUDE   COND=((3,3,CH,EQ,C'AAB'),&,
                ((17,7,CH,LT,333,7,CH),|,
                 (220,2,CH,EQ,C'RR')))
```

Analyzes as...

```
INCLUDE   COND=((3,3,CH,EQ,C'AAB'),&,((17,7,CH,LT,333,7,CH),|,(220,2,CH,EQ,C'RR')))
                 field   op literal    field   op  field       field   op  literal
```

Another example

```
OMIT   COND=(20,4,FI,GT,4096)
```

Introduction to the INREC Statement

```
INREC      BUILD=([separator,]position,length[,...])
```

❒ **Reformats the input records with only separator characters and the described fields**

- ♦ **That is, describe how to take input records and build the actual record to be sorted, copied, or merged**

- ♦ **The result of reformatting an input record is a "sort record" that contains data from the original record and various filler data (blanks, binary zeros, or literals); note: in this course, the term "sort record" can also mean records to be copied or merged**

  - ✗ So the BUILD operand describes how the sort record is constructed from a combination of separator characters and data from the input records

  - ✗ Note that FIELDS is a an older synonym for BUILD on both INREC and OUTREC statements

    - ➢ We will use BUILD in the lecture points, but you may find old jobs that use FIELDS; FIELDS should only be used in SORT, MERGE, and SUM statements

- ♦ **The sort records are built from byte one; the order in which data items are specified in the BUILD operand is the order they appear in the sort record, immediately adjacent to one another**

❒ **Punctuation, as shown, counts ...**

INREC Parameters

literal

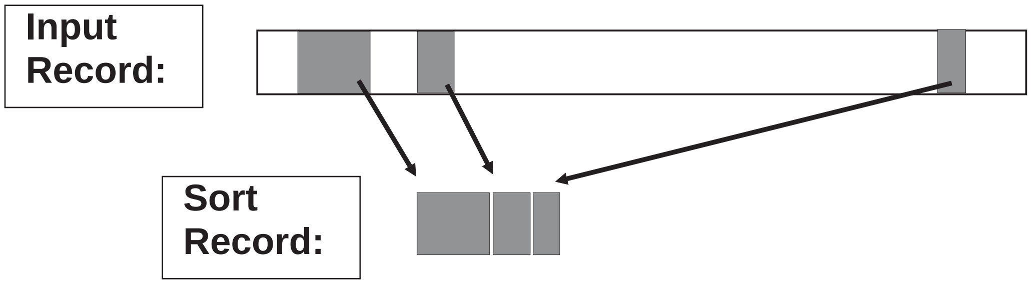
***n*X**	**—**	**insert *n* bytes of blanks (1 < *n* < 4095), for example**
		30X
***n*Z**	**—**	**insert *n* bytes of binary zeroes (1 < *n* < 4095),** *e.g.*
		26Z
***n*C'...'**	**—**	**insert *n* repetitions of the string (1<*n*<4095);**
or		
***n*X'...'**	**—**	**string is hex or character string up to 256 characters**

```
5C'*** HERE IT IS ***'
7X'80'
```

position,length

These two values are the actual starting position and length in the input record of the field to be extracted and put into the next available position in the "sort record"; you may build a sort record with no separators:

```
INREC   FIELDS=(5,20,37,3,99,3)
```

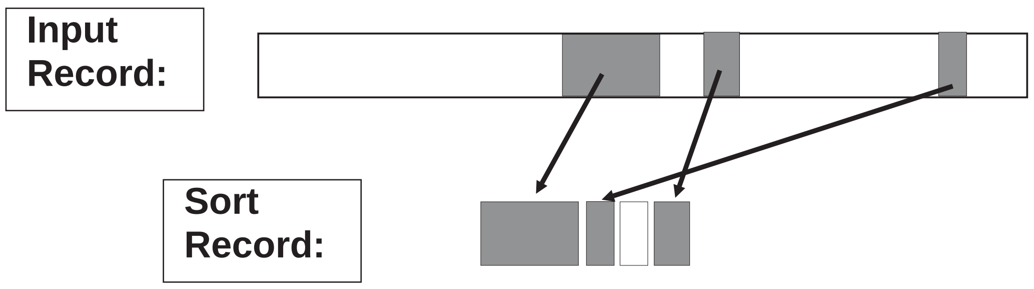


INREC Parameters, 2

Another example

- ♦ **Extract fields from an input record to build a record to be sorted, inserting some binary zeros to extend a packed decimal field:**

```
INREC       BUILD=(34,6,120,3,3Z,44,4)
```



❒ **The SORT statement FIELDS values (next page) describe positions in the "Sort Record", not the input record**

- ♦ **If your input record is variable length, be sure to allow for the 4-byte RDW - the first data byte is position 5**

Introduction to The SORT Statement

```
SORT      FIELDS=(position,length,format,sequence[,...])
```

♦ **Identify the field(s) to sort on, in decreasing order of importance**

```
SORT      FIELDS=(35,5,CH,A,3,2,PD,D,60,3,CH,A)
```

Notes on SORT FIELDS:

✗ All fields must be contained in the first 32752 bytes of the input records and the sum of the lengths of the control fields must be less than or equal to 4092 bytes

✗ If INREC is used to reformat input records, the *position* value is the position within the reformatted record

✗ Most common data formats are CH (character), BI (bit string), FI (fixed point signed), PD (packed decimal), and ZD (zoned decimal)

  ➢ After the data type, code A for ascending sequence or D for descending

✗ If you are sorting variable length records (including VSAM variable length records), the position values must allow for a 4 byte RDW at the front of the data

✗ Positions for BI data type can be specified on bit boundaries (*e.g.*: 5.3 is byte 5, bit 4); you may also specify lengths as some number of bytes and bits

Introduction to the OUTREC Statement

```
OUTREC      BUILD=([literal,]position,length[,...])
```

❒ **Reformats the sorted records on output**

- ♦ **Parameters have the same meaning and syntax as for INREC, but now formatting the final output records from the "sort record"s**

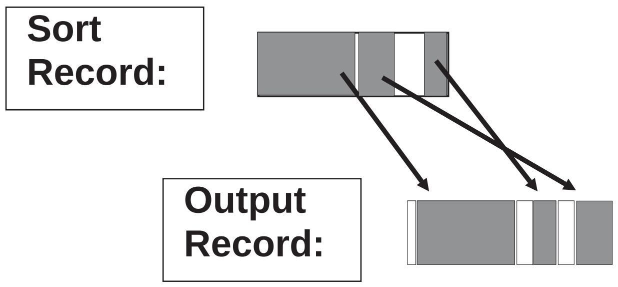
❒ **Nice for printed output especially**

Example

```
OUTREC         BUILD=(1X,1,30,2X,52,5,2X,31,10)
```

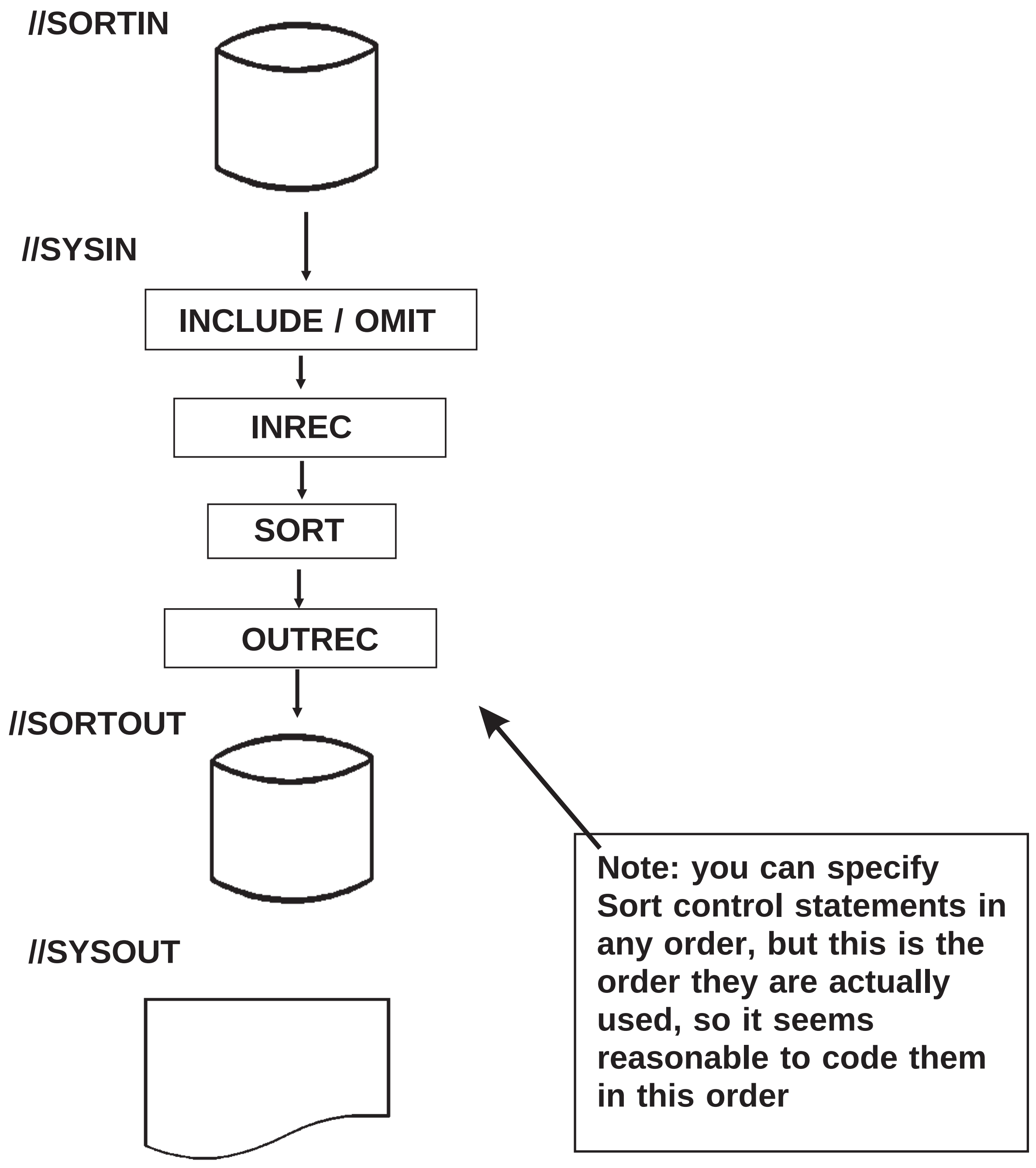
- ♦ **Picture: building output record from sort record**

1 blank
1-30 from sort record
2 blanks
52-56 from sort record
2 blanks
31-40 from sort record



❒ **Another application: inserting delimiters such as commas when passing the data to a program that handles such data (for example downloading to a PC database product that can handle comma-delimited input files)**

Control Statement Actions



Note: you can specify Sort control statements in any order, but this is the order they are actually used, so it seems reasonable to code them in this order

Sample SORT Steps

```
.
.
.
//STEP13       EXEC    PGM=SORT
//SORTIN       DD      DISP=SHR,DSN=GR55.TESTER.DATA
//SORTOUT      DD      DSN=GR55.ADDRESS,DISP=(,CATLG,DELETE),
//      SPACE=(260,(10,50)),AVGREC=K
//SYSIN        DD      *
        INCLUDE COND=...
        SORT BUILD=...
//SYSOUT       DD      SYSOUT=*
.
.
.
```

```
.
.
.
//STEP23       EXEC    PGM=SORT
//SORTIN       DD      DISP=SHR,DSN=AR55.RESTER.DATA
//SORTOUT      DD      DSN=AR55.PHONES,DISP=(,CATLG,DELETE),
//      LIKE=GR55.ADDRESS
//SYSIN        DD      DISP=SHR,DSN=AR55.PARMLIB(PHONES)
//SYSOUT       DD      SYSOUT=*
.
.
.
```

Using SORT To Do a Copy Operation

❒ **A COPY request may be made by …**

- ♦ **Specifying it on the SORT statement:**

```
SORT FIELDS=COPY
```

- ♦ **Or on the OPTION statement:**

```
OPTION COPY
```

❒ **When doing a COPY:**

- ♦ **INREC, OUTREC, INCLUDE, OMIT <u>may</u> be used (!)**
  - ✗ This could be very useful!

❒ **You can eliminate records with duplicate control key values by including this DFSORT control statement:**

```
SUM   FIELDS=NONE
```

- ♦ **Note that SUM will work for SORT and MERGE operations but not COPY operations**

Some Other Sort Capabilities

❒ **You can create multiple output files**

- ♦ **Reformatting records differently for different output files**
- ♦ **Splitting the records across different output files**

❒ **You can:**

- ♦ **Sort on two digit years (specify a century window through a run-time option and identify the date format from a list of options)**
- ♦ **Transform two-digit year dates to four-digit year dates**
- ♦ **Sort using locale processing**
    - ✗ Allowing DFSORT to be sensitive to collating sequences that differ between languages or cultural conventions
- ♦ **Use a symbol-specifications file pointed at by a SYMNAMES DD statement to sort on field names instead of offsets, lengths, and data type (different course ...)**
- ♦ **And lots more (different course...)**

Computer Exercise: SORT

Code the necessary JCL and SORT control statements to accomplish the following:

Sort the INPUTX inventory file (full name:____________________________)

Include only records that have a '5', '3', or '7' in the last position of Part Number

(INCLUDE statement)

Sort the file into ascending sequence by Old Part Number. The layout for records in INPUTX is on the following page.

(SORT statement)

The output of the sort should go to the printer.

BE SURE TO 'SHR' INPUTX

Exercise Stretch

Format the print lines:

(two blanks)**Old Part Number**(three blanks)**Part Number**(three blanks)**Description**

(OUTREC statement).

INPUTX Record Layout

Positions	Data
1 - 9	Part number; character
10 - 39	Description; character
40 - 44	Reserved; random character string
45 - 48	Unit Price; packed decimal: 9999V999
49 - 51	Quantity on hand; packed decimal: 99999
52 - 52	Reserved
53 - 54	Quantity on order; binary halfword; 999
55 - 56	Reorder level (also used as reorder quantity); binary halfword; 999
57 - 57	Switch; random bit string
58 - 66	Old Part Number; character
67 - 67	Reserved
68 - 77	Item Category; character
78 - 100	Reserved

Section Preview

❐ **OUTPUT Statements**

- ♦ **Introduction to OUTPUT statements**
- ♦ **Using OUTPUT statements (Machine Exercise)**

This page intentionally left almost blank.

Introduction to OUTPUT statements

❒ Suppose you have a job that produces a report for one of your users

- ♦ And that user shows the report in a meeting to make a point

❒ All of a sudden, three people in the room are interested in getting a copy of the report

- ♦ One wants it printed at her site, a remote location
- ♦ Another wants two copies, on his own local printer
- ♦ Another wants the report on microfiche

❒ What do you do?

- ♦ Have a programmer re-write the program to put the report to many DD statements? not likely
- ♦ Have a secretary or print services produce and distribute copies, including the microfiche and remote copies? Pretty expensive

❒ The best solution: use OUTPUT statements and solve the problem by making some small JCL changes ...

The OUTPUT Statement

❒ **The OUTPUT JCL statement allows you to process one or more SYSOUT data sets from a job in multiple ways**

❒ **Specify an OUTPUT statement for each collection of processing attributes you want to have available**

//name **OUTPUT** *paramete*r**s**

❒ **Then reference each appropriate OUTPUT statement on the DD statements for SYSOUT files you want processed that way**

//ddname **DD** **SYSOUT=*,OUTPUT=*.***name*

or

//ddname **DD** **SYSOUT=*,OUTPUT=(*.***name*$_1$**,*.***name*$_2$**,...)**

➢ If you reference multiple OUTPUT statements

♦ **Note: each OUTPUT JCL statement name must be unique within a <u>JOB</u>**

OUTPUT Statements: Examples

```
//GARGLE       JOB   ...
//FLOOR2     OUTPUT   DEST=PRT32,COPIES=2
//BLD91      OUTPUT   DEST=PRT56
//STEP1       EXEC    PGM=RPTS
//LIST1        DD     SYSOUT=A,OUTPUT=(*.FLOOR2,*.BLD91)
.
.
.
```

```
//GARGLE       JOB   ...
//FLOOR2     OUTPUT   DEST=PRT32,COPIES=2,DEFAULT=YES
//BLD91      OUTPUT   DEST=PRT56,DEFAULT=YES
//STEP1       EXEC    PGM=RPTS
//LIST1        DD     SYSOUT=A
.
.
.
```

```
//GARGOYLE    JOB       ...
//APPLMSGS   OUTPUT   CLASS=T,DEST=DENVER
//STEP1       EXEC    PGM=...
//COPY3      OUTPUT   COPIES=3,DEFAULT=YES
//ERRS         DD     SYSOUT=(,),OUTPUT=*.APPLMSGS
//RPT1         DD     SYSOUT=A
//RPT2         DD     SYSOUT=E,OUTPUT=*.APPLMSGS
.
.
.
//STEP2      EXEC     PGM=...
//RPTY      OUTPUT    CLASS=X,COPIES=2
//REMOTE    OUTPUT    DEST=RMT0003,COPIES=5
//RPTX        DD      SYSOUT=A,OUTPUT=*.REMOTE
//EXTRPT      DD      SYSOUT=(,),COPIES=3,
//      OUTPUT=(*.APPLMSGS,*.RPTY)
.
.
.
```

❒ **Note we are only showing SYSOUT type DD statements here**

OUTPUT Statement Parameters

❒ Many parameters on the OUTPUT statement are the same as for standard SYSOUT DD statements, although some are unique to OUTPUT

❒ Most useful OUTPUT parameters:

CLASS	=	SYSOUT class
COPIES	=	Number of original copies to produce
DEST	=	Route to local or remote printer(s)
PRTY	=	Priority in SYSOUT queue (0 - 255, lowest to highest)
JESDS	=	Indicates which JES datasets to include in this group (ALL, LOG, MSG, JCL); that is, how should JES data sets be processed?
DEFAULT	=	Should this set of parameters be a default for all SYSOUT datasets in the step this OUTPUT statement is placed in (or for the job if this statement is before any steps) (YES/NO)

❒ Note that the parameters CLASS, PRTY, JESDS, and DEFAULT are not DD statement parameters; they may only appear on OUTPUT statements

❒ Parameters on a SYSOUT DD statement override the same parameter on an OUTPUT statement

More OUTPUT Statement Parameters

❒ **You might find these parameters particularly useful**

- **NOTIFY=[***node***.]***userid***[...]**

    ✗ Notifies up to four user(s) that a print job has completed (if there is more than one entry in the list, put the list in parentheses)

    ✗ Note that this parameter only applies to certain types of printers and printer subsystems

- **OUTDISP=([***normal-disposition***][,***abnormal-disposition***]) (JES2 only)**

    ✗ How to dispose of an OUTPUT data set, especially if the job abends

    ✗ Options, for both parameters, are:

WRITE	**-**	**Print data out and purge**
HOLD	**-**	**Hold data until explicitly released**
KEEP	**-**	**Print but keep copy with LEAVE attribute**
LEAVE	**-**	**After release, change disposition to KEEP**
PURGE	**-**	**Delete without printing**

```
//SAVEOUT  OUTPUT  NOTIFY=$TRNCM,
//       OUTDISP=(KEEP,HOLD)
```

OUTPUT Statement Placement

- **An OUTPUT statement that appears before any EXEC statements is called a <u>job-level</u> OUTPUT statement**

  - ♦ **Only job-level OUTPUT statements may specify the JESDS parameter**

- **An OUTPUT statement that appears in a step is called a <u>step-level</u> OUTPUT statement**

- **An OUTPUT statement must appear physically prior to any SYSOUT DD statements that refer to it**

- **A job-level OUTPUT statement with DEFAULT=YES coded applies to all SYSOUT statements in the job**

  - ♦ **<u>Except</u> for SYSOUT statements in steps that contain step-level OUTPUT statements with DEFAULT=YES and for SYSOUT statements that have an OUTPUT= parameter coded**

- **A step-level OUTPUT statement with DEFAULT=YES coded applies to all SYSOUT statements in the step**

  - ♦ **<u>Except</u> for SYSOUT statements that have an OUTPUT= parameter coded**

- **The reference to a default OUTPUT statement is called <u>implicit</u> reference**

Explicit Reference To an OUTPUT Statement

❒ **A SYSOUT statement may explicitly reference any prior OUTPUT statement in the job by naming one or more OUTPUT statements in the 'OUTPUT' parameter:**

```
//REPT1      DD     SYSOUT=*,OUTPUT=*.GROUP1

//REPT42     DD     SYSOUT=A,OUTPUT=(*.GROUP1,*.OUT2)
```

❒ **Since parameters on DD statements take precedence over parameters on OUTPUT statements, to have the CLASS= parameter take effect, you need to code a null SYSOUT class on the DD statement:**

```
//EDITLST    DD     SYSOUT=(,),OUTPUT=(*.STEP5.TYPEAB)
```

❒ **If an OUTPUT JCL statement contains both JESDS and CLASS parameters, the CLASS parameter will override the MSGCLASS parameter on the JOB statement for the specified JES data set(s)**

❒ **References to an OUTPUT statement are of the form:**

.outputname*

OR ***.***stepname***.***outputname*

OR ***.***stepname***.***procstepname***.***outputname*

OUTPUT Statements: The Result

❒ For one of our earlier examples...

```
//GARGOYLE    JOB      ...
//APPLMSGS   OUTPUT   CLASS=T,DEST=DENVER
//STEP1       EXEC    PGM=...
//COPY3      OUTPUT   COPIES=3,DEFAULT=YES
//ERRS         DD     SYSOUT=(,),OUTPUT=*.APPLMSGS
//RPT1         DD     SYSOUT=A
//RPT2         DD     SYSOUT=E,OUTPUT=*.APPLMSGS
.
.
.
//STEP2      EXEC     PGM=...
//RPTY      OUTPUT    CLASS=X,COPIES=2
//REMOTE    OUTPUT    DEST=RMT0003,COPIES=5
//RPTX        DD      SYSOUT=A,OUTPUT=*.REMOTE
//EXTRPT      DD      SYSOUT=(,),COPIES=3,
//      OUTPUT=(*.APPLMSGS,*.RPTY)
.
.
.
```

SYSOUT Files Produced:

1. SYSOUT class T contains:
   1 copy of ERRS, routed to DENVER
   3 copies of EXTRPT routed to DENVER

2. SYSOUT class A contains:
   3 copies of RPT1
   5 copies of RPTX, routed to RMT0003

3. SYSOUT class E contains 1 copy of RPT2, routed to DENVER

4. SYSOUT class X contains 3 copies of EXTRPT

❒ HOW TO FIGURE: Take each applicable combination of SYSOUT statement and OUTPUT statement (options are not cumulative across groups of OUTPUT statements)

Solving the Original Problem

❒ **The report you showed to your colleagues was produced in this step:**

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING   DD   SYSOUT=H
```

❒ **How can we modify the step using OUTPUT statements to meet these requirements:**

- ♦ **One copy at DEST of SANDIEGO (use class of R)**
- ♦ **Two copies to DEST of LCL203 (use class H)**
- ♦ **One copy to microfiche (use class M)**

Solution JCL:

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
```

Solving the Original Problem

❒ The report you showed to your colleagues was produced in this step:

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING   DD   SYSOUT=H
```

❒ How can we modify the step using OUTPUT statements to meet these requirements:

- ♦ One copy at DEST of SANDIEGO (use class of R)
- ♦ Two copies to DEST of LCL203 (use class H)
- ♦ One copy to microfiche (use class M)

Solution JCL:

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//SANDIEGO OUTPUT DEST=SANDIEGO,CLASS=R
```

Solving the Original Problem

❐ The report you showed to your colleagues was produced in this step:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING    DD    SYSOUT=H
```

❐ How can we modify the step using OUTPUT statements to meet these requirements:

- ♦ One copy at DEST of SANDIEGO (use class of R)
- ♦ Two copies to DEST of LCL203 (use class H)
- ♦ One copy to microfiche (use class M)

Solution JCL:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//SANDIEGO OUTPUT DEST=SANDIEGO,CLASS=R
//LCL203   OUTPUT DEST=LCL203,COPIES=2,CLASS=H
```

Solving the Original Problem

❐ The report you showed to your colleagues was produced in this step:

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING   DD   SYSOUT=H
```

❐ How can we modify the step using OUTPUT statements to meet these requirements:

- ♦ One copy at DEST of SANDIEGO (use class of R)
- ♦ Two copies to DEST of LCL203 (use class H)
- ♦ One copy to microfiche (use class M)

Solution JCL:

```
//STEP12   EXEC  PGM=ASR44
//STEPLIB   DD   DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33    DD   DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//SANDIEGO OUTPUT DEST=SANDIEGO,CLASS=R
//LCL203   OUTPUT DEST=LCL203,COPIES=2,CLASS=H
//MICROF   OUTPUT CLASS=M
```

Solving the Original Problem

❐ The report you showed to your colleagues was produced in this step:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING    DD    SYSOUT=H
```

❐ How can we modify the step using OUTPUT statements to meet these requirements:

- ♦ One copy at DEST of SANDIEGO (use class of R)
- ♦ Two copies to DEST of LCL203 (use class H)
- ♦ One copy to microfiche (use class M)

Solution JCL:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//SANDIEGO OUTPUT DEST=SANDIEGO,CLASS=R
//LCL203   OUTPUT DEST=LCL203,COPIES=2,CLASS=H
//MICROF   OUTPUT CLASS=M
//ORIG     OUTPUT CLASS=H
```

Solving the Original Problem

❒ The report you showed to your colleagues was produced in this step:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//LISTING    DD    SYSOUT=H
```

❒ How can we modify the step using OUTPUT statements to meet these requirements:

- ♦ One copy at DEST of SANDIEGO (use class of R)
- ♦ Two copies to DEST of LCL203 (use class H)
- ♦ One copy to microfiche (use class M)

Solution JCL:

```
//STEP12    EXEC   PGM=ASR44
//STEPLIB    DD    DSN=HRD.PROD.LOADLIB,DISP=SHR
//MSTR33     DD    DSN=HRD.PRSNL.PEPLKSDS,DISP=SHR
//SANDIEGO OUTPUT DEST=SANDIEGO,CLASS=R
//LCL203   OUTPUT DEST=LCL203,COPIES=2,CLASS=H
//MICROF   OUTPUT CLASS=M
//ORIG     OUTPUT CLASS=H
//LISTING    DD   SYSOUT=(,),OUTPUT=(*.SANDIEGO,
//     *.LCL203,*.MICROF,*.ORIG)
```

Computer Exercise: OUTPUT Statements

Create a job to test using OUTPUT statements. Call this member JCLEX07 in your TR.CNTL data set.

Copy in the job that runs NEWF2F, member JCLEX04, (see page 456), where the last step puts a copy of our file to SYSOUT, or the SORT exercise (see page 517).

Modify the job so that printed output from the SYSOUT file goes to two or more TSO output classes. To ensure data stays around, use OUTDISP=(HOLD,HOLD) for the second output.

Check that this works using SDSF, IOF, (E)JES, or FLASHER as proof.

Exercise Stretch: test using DEFAULT=YES to pick up files that don't reference any OUTPUT statements explicitly; experiment with job-level OUTPUT statements to route all or just some of the JES data sets.

Section Preview

❒ **Condition Code Testing and Memory Management**

- ♦ **REGION parameter**
- ♦ **MEMLIMIT parameter**
- ♦ **Program Termination**
- ♦ **IF / THEN / ELSE / ENDIF statements**
- ♦ **JOBRC parameter**
- ♦ **Conditional Processing (Machine Exercise)**

The REGION Parameter

❒ **Not all possible virtual storage in an address space is made available to any job or step automatically**

- ♦ **This reduces the overhead of managing page tables**

❒ **The amount of virtual storage made available beyond your initial program size is supplied as an installation-chosen default region size**

❒ **As your program executes, it may make requests for virtual storage, either explicitly or implicitly**

❒ **These dynamic storage requests are satisfied from your region amount**

❒ **Should you need more virtual storage than available, your program ABENDs with a system completion code of 80A**

❒ **To override the default region size, you may specify the REGION parameter:**

REGION=_nnnn_{**K|M**}

For Example

```
REGION=512K
REGION=200M
```

❒ **This parameter may be specified on the JOB or EXEC statements**

The MEMLIMIT Parameter

- ❐ **In the same way REGION controls use of storage above the line, MEMLIMIT controls access to storage above the bar**

  - ♦ **Note that REGION only controls storage below the bar**

Syntax

MEMLIMIT={NOLIMIT|*n*{M|G|T|P}}

Examples

```
MEMLIMIT=NOLIMIT

MEMLIMIT=4M

MEMLIMIT=1000G
```

- ❐ **MEMLIMIT may be coded on the JOB statement or an EXEC statement**

  - ♦ **This may be further restricted by an installation exit**

Program Termination

Normal Termination	**Abnormal Termination**
CONDITION CODE or **RETURN CODE** **Value: 0 - 4095**	**COMPLETION CODE** one of: **User Completion Code: U*dddd*** **System Completion Code: S*xxx*** **z/OS supplies SCC**

	User specifies RETURN CODE by:	**User specifies UCC BY:**
LE (all languages)	**Call to CEE3SRC**	**Call to CEE3ABD**
ALC:	**Value in R15 at RETURN time**	**ABEND *dddd***
COBOL:	**Value in RETURN-CODE special register**	**CALL 'ILBOABN0' USING identifier (where "identifier" has value of *dddd* and attributes PIC S9999 COMP)**
PL/I:	**Issue CALL PLIRETC(*arg*) before ending program**	**Requires installation modification of IBM-supplied module IBMBEER**
C:	**Value in** return() **or** exit()	**Call to CEE3ABD**

IF / THEN, ELSE, ENDIF Statements

❐ **These statements work together in a set, as follows**

```
//[name]   IF   relational-expression   THEN
.
.         action if relational-expression is true
.
//[name]   ELSE
.
.         action if relational-expression is false
.
//[name]   ENDIF
```

<u>**Notes**</u>

♦ **ELSE is optional, ENDIF is required**

♦ **The statements under either the THEN clause or the ELSE clause may be**

✗ <u>Omitted</u> (a null action) (both can't be null)

✗ <u>One or more steps</u> (EXEC statements for programs or procedures with associated DD, OUTPUT, and other related JCL statements)

> ➢ This construct does not conditionally control the processing of JCL; it conditionally controls the execution of job steps

✗ <u>Nested IF/THEN, ELSE, ENDIF statements</u> (up to 15 levels deep)

IF / THEN - Relational Expressions

❒ **Relational expressions on IF / THEN statements consist of**

- **Relational expression keywords**

  RC - Return Code

  ABEND - Abend condition

  ¬ABEND - No abend condition occurred

  ABENDCC - Specific system or user Abend completion code

  RUN - Step ran

  ¬RUN - Step did not run

- **Comparison operators (may use either alphabetic abbreviation or symbol)**

GT	>	(for greater than)
LT	<	(for less than)
NG	¬>	(for not greater than)
NL	¬<	(for not less than)
EQ	=	(for equal)
NE	¬=	(for not equal)
GE	>=	(for greater than or equal)
LE	<=	(for less than or equal)

- **Logical operators (use either the words or the symbols)**

AND	&
OR	|
NOT	¬

IF / THEN - Relational Expressions, 2

Examples

```
//STEPPER  IF  RC > 4        THEN
//STEPPER  IF  (RC > 4)      THEN
```

Parentheses optional

- ♦ If you don't specify a step name on RC, RC refers to the greatest return code from any step so far

```
//TSO23F  JOB  ------------
//STEP1   EXEC  PROC=ISDEDIT1
//         IF   RC GT 4  THEN
//ALT1    EXEC  PROC=SETUP1
//ALT1A   EXEC  PROC=DO1
//        ELSE
//ALT2    EXEC  PROC=SETUP2
//ALT2A   EXEC  PROC=DO2
//        ENDIF
//...
```

Use abbreviation (GT) instead of symbol

- ♦ Names are optional on IF, ELSE, and ENDIF statements

IF / THEN - Relational Expressions, 3

Examples, continued

```
//STOPPER  IF  STEP5.RC > 8  THEN
```

Example of stepname qualification

```
//U3456    JOB   ----------
//STEP1    EXEC  PGM=ARGDPT2
//INDD      DD   ------
//MASTDD    DD   ------
. . .
//          IF  STEP5.RC > 8  THEN
//STEP6A   EXEC  PGM=NOONE1,PARM=ANYTIME
//ITSDD     DD   ----------
//MYDD      DD   ----------
//         ENDIF
```

- ♦ **If you specify a step name on any of the keywords, if the step was not run the result of the test is always false**

❐ **Spaces are optional before and after special symbol operators (such as = and ¬=), but required before and after alphabetic operators (*e.g.*: EQ and NE)**

- ♦ **Spaces are required around the & and | operators**

IF / THEN - Relational Expressions, 4

Examples, continued

Examples of compound expressions

```
//STIFF  IF  ((RC < 12 & RC NE 7) | (RC = 13)) THEN
```

```
//STIFF  IF  ((RC < 12 & RC NE 7) |
//          (RC = 13))           THEN
```

Example of continuation

```
//TOEHOLD  JOB  --------------
//ONE      EXEC  PGM=WHOOSIS
//WHODD1    DD   ----------
//WHODD2    DD   ----------
.
.
.
```

Example using continuation and alphabetic connectives

```
//          IF   ((RC < 16 OR THREE.RC NE 7)
//            AND (ONE.RC <= 4))  THEN
//LATER    EXEC  PROC=GATOR
//EATER    EXEC  PROC=FEASTER
//         ELSE
//WAITER   EXEC  PROC=THINKER
//         ENDIF
```

IF / THEN - Relational Expressions, 5

Examples, continued

Examples of RUN

```
//TRAPPER  IF  STEP7.RUN=TRUE  THEN
//TRAPPER  IF  STEP7.RUN       THEN


//TRIPPER  IF  ¬ONE.RUN        THEN
```

♦ The RUN keyword requires a stepname

Examples of ABEND

```
//STAMPER  IF  ABEND=TRUE    THEN
//STAMPER  IF  ABEND         THEN

//STOMPER  IF  ABEND=FALSE   THEN
//STOMPER  IF  ¬ABEND        THEN

//SUFFER   IF  STEP5.ABEND   THEN
```

♦ If you omit a step name on ABEND or ¬ABEND, the test is for any previous step

IF / THEN - Relational Expressions, 6

Examples, continued

Example of ABEND codes

```
//STAFFER  IF  ABENDCC=S013      THEN
//STOUFER  IF  ABENDCC ¬= U0102  THEN


//WHOPPER  IF  STEP2.ABENDCC = U0050  THEN
```

- **If you omit a step name on ABEND or ¬ABEND, the test is for any previous step**
- **If you omit a step name on ABENDCC, the test for a value applies to the most recent abend code**
- **System completion codes are specified as an 'S' followed by three hex digits**
- **User completion codes are specified as a 'U' followed by four decimal digits (0000-4095)**

```
//         IF  (ABEND | RC > 16 | ¬STEP5.RUN)
//         THEN
//LOSER      EXEC  FIXER
//         ELSE
//LASTER     EXEC  WRAPUP
//         ENDIF
```

IF / THEN - Relational Expressions, 7

Examples, concluded

- **A pathological example using nested IFs**

```
//WHOA23   JOB  -----------
//STEP1    EXEC  PGM=ANYONE
//ANY1DD1   DD   ----------
. . .
//         IF  (STEP1.RC = 5) THEN
//           IF  (STEP2.RC > 4) THEN
//             IF  (STEP5.RC = 0 & ¬ABEND) THEN
//CHOICE1  EXEC  PROC=ANYTWO
//             ELSE
//CHOICE2A EXEC  PROC=MANYTWO
//             ENDIF
//           ELSE
//             IF  (STEP5.RC > 0 | ABEND) THEN
//CLEAN22   EXEC  PROC=CLEANER,PARM.A22='NO WAY'
//LASTCL    EXEC  PGM=LASTT,PARM='VOL23'
//             ENDIF
//           ENDIF
//HUNTER    EXEC  PROC=SOMETIME
//         ENDIF
```

IF / THEN - Relational Expressions, 8

Notes

- **If you specify a step name on any of the keywords, if the step was not run the result of the test is always false**

- **If an abend occurs and there is no testing of the ABEND condition for a step, that step will not run**

- **Certain situations will terminate the remainder of a job regardless of any testing**

    - ✗ Job TIME expires

    - ✗ Referenced data set was not created or cataloged in a previous step

    - ✗ Abnormal termination during step scheduling, such as JCL errors or inability to allocate DASD space

❐ **NOTE: an earlier JCL parameter, COND, is still found in JCL**

- **It is less powerful than the IF construct, but many people are familiar (and comfortable) with this feature**

- **If you have a need to understand this parameter, check out the section starting on page 623**

The JOBRC Parameter

❒ **In z/OS 1.13, a new parameter was introduced for the JOB statement: JOBRC**

Syntax

JOBRC={MAXRC|LASTRC|(STEP,*stepname***[.***procstepname***])}**

Where ...

- **MAXRC means the return code for the job should be largest condition code from any step or the last ABEND code from any step - this is the default**

- **LASTRC means the return code for the job should be the return code or ABEND code from the last step that was run**

- **(STEP,***stepname***[.***procstepname***]) means the return code from the job should be the return code or ABEND code from the specified step (or step within procedure - discussed shortly)**

Examples

```
//WILLY  JOB  ,,JOBRC=LASTRC

//NILLY  JOB  ,,JOBRC=(STEP,STEP7)
```

Computer Exercise: Conditional Processing

1. Build a job based on the earlier exercise JCLEX04 (p. 456). So, edit new member JCLEX08 and instead of copying JOB to begin, copy JCLEX04. Then modify the JCL as described below.

    Note that NEWF2F accepts a PARM value, any 0 to 25 characters. If the length is greater than 0, the PARM data is written to the JCL listing. In any case, the length of the PARM data is used as the condition code value for the program(!).

2. For the first step, pass a PARM value of WHATEVER. For the OUTDD data set, code SYSOUT=*. Leave the INDD statement as is.

3. For the second step, pass a PARM value of WHAT. For the INDD data set, use __________TRAIN.INPUTX [BE SURE YOU HAVE DISP=SHR!]. For the OUTDD data set, code SYSOUT=*.

4. For the third step, pass a PARM value of HAT. For the INDD data set, point to member INPUT2 in your TR.CNTL library [BE SURE THAT YOU HAVE DISP=SHR!]. For the OUTDD data set, code SYSOUT=*,

5. Use IF / THEN / ELSE / ENDIF so that if the return code from step 1 is 8 run step 2, else run step 3.

6. Run this job. Which step was executed, step 2 or step 3?

Hint: sometimes it helps to first just code the job and run it with no conditional processing, to look at the outputs (especially note the condition codes).

This page intentionally left almost blank.

Section Preview

❐ **JCL Procedures**

- ♦ **Cataloged procedures**
- ♦ **JCLLIB Statement**
- ♦ **A Cataloged Procedure (Machine Exercise)**

Cataloged Procedures

❒ **A <u>cataloged procedure</u> is a collection of pre-coded JCL**

- **Stored as a member in SYS1.PROCLIB or some similar PDS**
- **Standardizes frequently used JCL coding**
- **Reduces keying effort with JCL**
- **May contain EXEC, DD, OUTPUT, Comment, IF/THEN, ELSE, ENDIF, and certain other statements we haven't even talked about yet**
- **May not contain:**

  JOB statements

  JOBLIB DD statements

  SYSIN data ('DD *' or 'DD DATA')

Special note: from z/OS 1.13, JES2 systems <u>can</u> have SYSIN data inside a procedure

❒ **A cataloged procedure is invoked from a job stream using an EXEC statement:**

```
        //STEPNM      EXEC      PROC=procname
or
        //STEPNM      EXEC      procname
```

JCLLIB

❐ **The JCLLIB statement identifies private libraries the system should use to search for**

- **Cataloged procedures**
- **INCLUDE groups (discussed later)**

Syntax

//[name]   JCLLIB   ORDER=(*library*[,*library*...])

Notes

- **The order the libraries are specified is the order they are searched**
- **If a procedure or INCLUDE group is not found in any of the libraries, the system procedure library (SYS1.PROCLIB and its concatenations) is searched**
- **You can code only one JCLLIB statement per job**
- **Must appear after the JOB statement and before any EXEC or INCLUDE statements**
- **May not appear in an INCLUDE group**

Example

```
//MYPROC  JCLLIB  ORDER=(DEPT53.PROD.PROCLIB,
//    DEPT53.NEW.PROCLIB)
```

Cataloged Procedures, continued

❒ Suppose we have a member called RRLD23 in a private proclib (DEPTRLD.PROCLIB), and the member looks like this:

```
//*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
//*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
//STEP1   EXEC  PGM=IRLD11,PARM='DES MOINES'
//TIMESLIP DD   DSN=HRD.DM.TIMESLPS,DISP=SHR
//EMPFILE  DD   DSN=HRD.DM.EMPLYS,DISP=OLD
//BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
//         DD   DSN=HRD.DM.BENEFITS,DISP=SHR
//RAWCHKS  DD   DSN=HRD.DM.RAWCHKS,DISP=(,PASS),
//   SPACE=(120,(1000,1000)),BLKSIZE=0,AVGREC=K
//SUMRY    DD   SYSOUT=E
//STEP2   EXEC  PGM=PRCHKS,PARM='DES MOINES'
//INRAW    DD   DSN=HRD.DM.RAWCHKS,
//    DISP=(OLD,DELETE)
//CHECKS   DD   SYSOUT=V
```

❒ Then, we can invoke this procedure by submitting the following JCL:

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//       JCLLIB  ORDER=(DEPTRLD.PROCLIB)
//RUNIT    EXEC  RRLD23
```

Cataloged Procedures, continued

❒ **On your listing you would see the following:**

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//       JCLLIB  ORDER=(DEPTRLD.PROCLIB)
//RUNIT    EXEC  RRLD23
XX*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
XX*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
XXSTEP1   EXEC  PGM=IRLD11,PARM='DES MOINES'
XXTIMESLIP DD   DSN=HRD.DM.TIMESLPS,DISP=SHR
XXEMPFILE  DD   DSN=HRD.DM.EMPLYS,DISP=OLD
XXBENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
XX         DD   DSN=HRD.DM.BENEFITS,DISP=SHR
XXRAWCHKS  DD   DSN=HRD.DM.RAWCHKS,DISP=(,PASS),
XX   SPACE=(120,(1000,1000)),BLKSIZE=0,AVGREC=K
XXSUMRY    DD   SYSOUT=E
XXSTEP2   EXEC  PGM=PRCHKS,PARM='DES MOINES'
XXINRAW    DD   DSN=HRD.DM.RAWCHKS,
XX    DISP=(OLD,DELETE)
XXCHECKS   DD   SYSOUT=V
```

❒ **The Converter has merged the cataloged procedure with the JCL**

- **On your listing, it shows JCL from a procedure with 'XX' in columns 1 and 2 instead of '//'**

Testing Procedures

- ❒ You can see, even with this small procedure, that coding procedures is as complicated as coding regular JCL
  - ♦ The potential for error is high

- ❒ Before placing a procedure in SYS1.PROCLIB (which is a highly controlled data set), or a production proclib, you need to test a procedure thoroughly
  - ♦ This is what private procedure libraries are for

Computer Exercise: A Cataloged Procedure

Construct a member, named COPYNPRT, in your TR.CNTL library that is a two step procedure, as follows:

Step One

Run a program called NEWF2F. This program is found in the library ________________TRAIN.LOADLIB. This program takes two DD statements: INDD and OUTDD; use this information ...

INDD should point to __________TRAIN.INPUTA (DISP=SHR, please)

OUTDD should point to a new temporary disk data set (two tracks of space is plenty); PASS this data set to the next step.

Step Two

Run a program called SORT. This program requires four DD statements: SORTIN, SORTOUT, SYSIN, SYSOUT; use this information ...

SORTIN should point to the disk data set created in Step One above; delete this file at end of step

SORTOUT should go to a printer

SYSIN should point to member CNTRL3 in the data set __________TRAIN.LIBRARY

SYSOUT should go to a printer.

Construct another member in TR.CNTL, JCLEX09, that is a job that includes:

A JOB statement [and any necessary JES statements]
A JCLLIB statement to point to your TR.CNTL library
An EXEC statement to invoke your procedure COPYNPRT

Run this job.

This page intentionally left almost blank.

Section Preview

❐ **JCL Procedures: Inserts and Overrides**

- ♦ **Procedures and Inserts**
- ♦ **Procedures and Overrides**
- ♦ **Inserts and Overrides (Machine Exercise)**

Procedures and Inserts

❐ Procedures wouldn't be used much if this is all you could do with them

- ♦ But they are really very flexible ...

❐ For example, wouldn't it be nice if you could change which data set you were using for input each time you ran the proc?

- ♦ You can do this if you code the proc as more of a skeleton and omit DD statements that you want to add at run time

❐ Suppose, using our RRLD23 example, we needed to run this proc once each pay period for each location

- ♦ Each time we run the proc, the TIMESLIP DD statement has to point to the time slips for the employees at a different location
- ♦ The solution: don't include TIMESLIP in the proc, but add the appropriate statement at run time
- ♦ At run time we will insert the correct DD statement in the correct location ...

Procedures and Inserts, continued

❒ Our proc (now assumed to be in SYS1.PROCLIB, for simplicity):

```
//*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
//*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
//STEP1   EXEC  PGM=IRLD11,PARM='DES MOINES'
//EMPFILE  DD   DSN=HRD.DM.EMPLYS,DISP=OLD
//BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
//         DD   DSN=HRD.DM.BENEFITS,DISP=SHR
//RAWCHKS  DD   DSN=HRD.DM.RAWCHKS,DISP=(,PASS),
//   SPACE=(120,(1000,1000)),BLKSIZE=0,AVGREC=K
//SUMRY    DD   SYSOUT=E
//STEP2   EXEC  PGM=PRCHKS,PARM='DES MOINES'
//INRAW    DD   DSN=HRD.DM.RAWCHKS,
//    DISP=(OLD,DELETE)
//CHECKS   DD   SYSOUT=V
```

❒ When we run this, after the EXEC statement that invokes the procedure we add any DD statements we wish to insert, using the special notation

//stepname.ddname   DD   ---

- Where <u>stepname</u> designates the step the DD statement belongs to

- In our case, we want to insert a DD statement named TIMESLIP into the STEP1 step ...

Procedures and Inserts, continued

❒ Using this notation, our job to run this proc might look like this:

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23
//STEP1.TIMESLIP  DD  DSN=HRD.DM.TIMESLPS,
//    DISP=SHR
```

❒ On the JCL listing, we would see something like this:

```
 1 //ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
   //    MSGCLASS=A,NOTIFY=LOGGER
 2 //RUNIT    EXEC  RRLD23
   XX*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
   XX*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
 3 XXSTEP1   EXEC  PGM=IRLD11,PARM='DES MOINES'
 4 XXEMPFILE  DD   DSN=HRD.DM.EMPLYS,DISP=OLD
 5 XXBENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
   XX         DD   DSN=HRD.DM.BENEFITS,DISP=SHR
 6 XXRAWCHKS  DD   DSN=HRD.DM.RAWCHKS,DISP=(,PASS),
   XX   SPACE=(120,(1000,1000)),BLKSIZE=0,AVGREC=K
 7 XXSUMRY    DD   SYSOUT=E
 8 //STEP1.TIMESLIP DD   DSN=HRD.DM.TIMESLPS,
   //    DISP=SHR
 9 XXSTEP2   EXEC  PGM=PRCHKS,PARM='DES MOINES'
10 XXINRAW    DD   DSN=HRD.DM.RAWCHKS,
   XX    DISP=(OLD,DELETE)
11 XXCHECKS   DD   SYSOUT=V
```

- Notice that the inserted DD statement appears at the end of the step it is being inserted into

Procedures and Inserts, continued

❐ **You can insert any number of DD statements into any number of steps**

- ♦ **It used to be that you had to code your inserted DD statements in step-order, matching the order of the steps in the procedure**

  - ✗ This is no longer true: but it is a good idea

❐ **You can also use inserts as a way to pass SYSIN type data to a step in a proc:**

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23
//STEP1.TIMESLIP  DD  *
01223400000350000222
01224400106650001654
01231100029500013666
.
.
.
```

Procedures and Overrides

❒ You can override existing DD statements

- Use the “stepname.ddname” syntax, only specifying the parameters you wish to override

- Non-overridden parameters will apply as they are coded in the proc

❒ Use overrides to change parameters for specific invocations

- For example, using our original proc, ...

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23
//STEP1.BENTABLE  DD
//                DD  DSN=TEST.NEW.BENEFITS
//STEP1.RAWCHKS   DD  SPACE=(120,(5000,5000))
//STEP1.SUMRY     DD  COPIES=3
//STEP2.CHECKS    DD  SYSOUT=W
```

Procedures and Overrides, continued

❐ **At run time, the JCL listing would show something like this**

```
 1 //ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
   //    MSGCLASS=A,NOTIFY=LOGGER
 2 //RUNIT   EXEC  RRLD23
   XX*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
   XX*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
 3 XXSTEP1   EXEC  PGM=IRLD11,PARM='DES MOINES'
 4 XXTIMESLIP DD   DSN=HRD.DM.TIMESLPS,DISP=SHR
 5 XXEMPFILE  DD   DSN=HRD.DM.EMPLYS,DISP=OLD
 6 //STEP1.BENTABLE  DD
   X/BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
 7 //               DD  DSN=TEST.NEW.BENEFITS
   X/         DD   DSN=HRD.DM.BENEFITS,DISP=SHR
 8 //STEP1.RAWCHKS  DD  SPACE=(120,(5000,5000))
   X/RAWCHKS  DD   DSN=HRD.DM.RAWCHKS,DISP=(,PASS),
   X/   SPACE=(120,(1000,1000)),BLKSIZE=0,AVGREC=K
 9 //STEP1.SUMRY  DD  COPIES=3
   X/SUMRY    DD   SYSOUT=E
10 XXSTEP2   EXEC  PGM=PRCHKS,PARM='DES MOINES'
11 XXINRAW    DD   DSN=HRD.DM.RAWCHKS,
   XX    DISP=(OLD,DELETE)
12 //STEP2.CHECKS  DD  SYSOUT=W
   X/CHECKS   DD   SYSOUT=V
```

❐ **Notice that all the overrides are placed just before the statement they are overriding**

- ♦ **The overridden statement shows with 'X/' in columns 1 and 2**

❐ **Overrides should be in step-order, and within a step, overrides should be in the same relative order as the DD statements**

- ♦ **This is not strictly required, but the discipline is good**

Procedures and Overrides, continued

❐ **Note how the concatenated data sets JCL was used to position to the correct DD statement(s) in the list to override**

- ♦ **This will work with adds, too, to concatenate a new data set to an existing one:**

```
//STEP1.TIMESLIP  DD
//                DD  DSN=HRD.YSTRDY.TIMES,DISP=SHR
```

❐ **DD statement overrides and inserts may be done in a single invocation:**

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23
//STEP1.TIMESLIP  DD  DSN=HRD.DM.TIMESLPS,
//    DISP=SHR
//STEP1.BENTABLE  DD
//                DD  DSN=TEST.NEW.BENEFITS
//STEP1.RAWCHKS   DD  SPACE=(120,(5000,5000))
//STEP1.SUMRY     DD  COPIES=3
//STEP2.CHECKS    DD  SYSOUT=W
//STEP2.SYSUDUMP  DD  SYSOUT=*
```

Overrides on EXEC Statements

❒ You may override and insert parameters on any EXEC statements in a cataloged procedure:

- ♦ Specify on the EXEC statement that invokes the procedure,

```
parametername.stepname=value
```

- ♦ Just the opposite of the way we do it for the DD statements!

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23,PARM.STEP1='DETROIT',
//   REGION.STEP1=1M,MEMLIMIT.STEP2=100G
```

❒ Parameters should be specified in step order (but not required)

❒ You may not override the PGM parameter

❒ If you specify a parameter with no 'stepname', the value applies to all steps in the procedure (except the PARM parameter: this only applies to the first step)

Additional Overrides and Inserts

❒ You may override or insert OUTPUT statements in a procedure

❒ If you specify a DD statement with no 'stepname', that statement is associated with the step of the previous DD statement; if there is no previous DD statement, it is associated with the first step

❒ You may nullify parameters on EXEC, DD, or OUTPUT statements by coding the parameter as 'parm=' with no following value:

```
REGION.STEP1=,PARM.STEP1='CHICAGO',MEMLIMIT.STEP2=
```

- ♦ In this example, we nullify any REGION parameter on STEP1, override the PARM parameter on STEP1, and nullify any MEMLIMIT test on STEP2

❒ If you have a step after an invocation of a procedure, you may refer back to a DD statement in the procedure using "DSN=*.stepname.procstepname.ddname"

- ♦ Where "stepname" is the name on the EXEC statement invoking the proc and "procstepname" is the name on the EXEC statement in the proc itself

For example

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23,PARM.STEP1='DETROIT',
//   REGION.STEP1=1M,MEMLIMIT.STEP2=5G
//XTRA     EXEC  PGM=PARTS2
//RDINP     DD   DSN=*.RUNIT.STEP1.TIMESLIP,DISP=SHR
//RPTOP     DD   SYSOUT=*
```

Computer Exercise: Inserts and Overrides

Create member JCLEX10 that is a copy of JCLEX09, your existing job that runs COPYNPRT, and modify JCLEX10 as follows:

1. Add a statement to concatenate a data set to the INDD DD statement in Step One; (after the EXEC statement that invokes the proc, code the necessary JCL to concatenate ________.TRAIN.INPUTX to INPUTA (hint: this takes two lines); do not modify the proc.) Note that this is an **override** of your proc.

   Run the job this way.

2. In COPYNPRT, delete the INDD statement from Step One; modify JCLEX10 so there is a single DD statement named INDD inserted in Step One; this should simply point to INPUTA; note that the INDD in your job that invokes COPYNPRT is now an **insert** to your proc.

   Run the job this way.

3. Modify JCLEX10 so that INDD now points to ________.TRAIN.INPUTX instead of ________.TRAIN.INPUTA. Do not modify the proc, just type over the DSN on the INDD insert.

   (Note: this is a one-character change.)

   Run the job this way.

4. Modify JCLEX10 so that INDD points to _______.TRAIN.LIBRARY(INPUTC); in the SORT step the SYSIN DD statement should point to member CNTRL2 in the LIBRARY data set. Do not modify the proc, just use overrides and inserts as necessary.

   Run the job this way.

This page intentionally left almost blank.

Section Preview

❐ **Symbolic Parameters**

- ♦ **Symbolic Parameters**
- ♦ **SYSUID**
- ♦ **A Procedure With Symbolic Parameters (Machine Exercise)**

Procedures And Symbolic Parameters

- ❒ **When you know you will be overriding parameters on JCL in a procedure almost every time, you may prefer to code the procedure using <u>symbolic parameters</u>**

- ❒ **A symbolic parameter is a variable that you intend to replace with a value at run time, thus deferring the decision until necessary**

- ❒ **When you invoke the procedure, you specify values for these parameters on the EXEC statement that invokes the procedure**

- ❒ **In a procedure, a symbolic variable is written as an ampersand (&) followed by 1 to 8 alphanumeric or national characters:**

```
//STEP1    EXEC  PGM=IRLD11,PARM=&LOCATION
```

- ❒ **When you invoke the procedure, you specify values by naming the symbolic parameter <u>without</u> the ampersand, followed by an equals sign (=) and the value:**

```
//DOIT     EXEC  RRLD23,LOCATION='CHICAGO'
```

- ❒ **Then, everywhere in the procedure that '&LOCATION' appears, it is replaced by the value assigned to LOCATION**

A Procedure With Symbolic Parameters

❒ **Here's our procedure using some symbolic parameters:**

```
//*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
//*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
//STEP1   EXEC  PGM=IRLD11,PARM=&LOCATION
//TIMESLIP DD   DSN=HRD.&LOC..TIMESLPS,DISP=SHR
//EMPFILE  DD   DSN=HRD.&LOC..EMPLYS,DISP=OLD
//BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
//         DD   DSN=HRD.&LOC..BENEFITS,DISP=SHR
//RAWCHKS  DD   DSN=HRD.&LOC..RAWCHKS,DISP=(,PASS),
//   SPACE=(120,(&PRI,&SEC)),BLKSIZE=0,AVGREC=K
//SUMRY    DD   SYSOUT=&SUMRY
//STEP2   EXEC  PGM=PRCHKS,PARM=&LOCATION
//INRAW    DD   DSN=HRD.&LOC..RAWCHKS,
//    DISP=(OLD,DELETE)
//CHECKS   DD   SYSOUT=V
```

❒ **Notice that symbolic parameters may appear on any statement (EXEC, DD, OUTPUT, etc.)**

- **You can even override the PGM value with this**
- **Symbolic substitution does not take place on comments or for name fields**

Invoking A Procedure With Symbolic Parameters

❒ Thus, a job to run our proc using symbolics might look like this

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23,LOCATION=('LOS ANGELES'),
//    LOC=LA,PRI=1000,SEC=500,SUMRY='*'
//STEP1.BENTABLE  DD
//                DD  DSN=TEST.&LOC..BENEFITS
//STEP1.SUMRY     DD  COPIES=3
```

❒ Note that special characters need to be enclosed in single quotes (apostrophes), and that if the special characters include blanks, the whole value needs to be bounded by parentheses

- ♦ The apostrophes are not part of the resulting value, nor are the parentheses
- ♦ The parentheses are not needed, even if the value contains blanks, if the value is hard coded in the proc

❒ Overrides and inserts may reference symbolics in the procedure

❒ Symbolics and standard overrides may be combined on the EXEC statement

❒ Invoking a procedure using symbolics may also be combined with DD statement overrides and inserts

Invoking A Procedure With Symbolic Parameters, 2

❒ **Your JCL listing will look something like this:**

```
 1 //ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
   //    MSGCLASS=A,NOTIFY=LOGGER
 2 //RUNIT   EXEC  RRLD23,LOCATION=('LOS ANGELES'),
   //    LOC=LA,PRI=1000,SEC=500,SUMRY='*'
   XX*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
   XX*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
 3 XXSTEP1   EXEC  PGM=IRLD11,PARM=&LOCATION
   IEFC653I SUBSTITUTION JCL - PGM=IRLD11,PARM=('LOS ANGELES')
 4 XXTIMESLIP DD   DSN=HRD.&LOC..TIMESLPS,DISP=SHR
   IEFC653I SUBSTITUTION JCL - DSN=HRD.LA.TIMESLPS,DISP=SHR
 5 XXEMPFILE  DD   DSN=HRD.&LOC..EMPLYS,DISP=OLD
   IEFC653I SUBSTITUTION JCL - DSN=HRD.LA.EMPLYS,DISP=OLD
 6 //STEP1.BENTABLE  DD
   X/BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
 7 //         DD   DSN=TEST.&LOC..BENEFITS
   IEFC653I SUBSTITUTION JCL - DSN=TEST.LA.BENEFITS
   X/               DD  DSN=HRD.&LOC..BENEFITS,DISP=SHR
   IEFC653I SUBSTITUTION JCL - DSN=HRD.LA.BENEFITS,DISP=SHR
 8 XXRAWCHKS  DD   DSN=HRD.&LOC..RAWCHKS,DISP=(,PASS),
   XX   SPACE=(120,(&PRI,&SEC)),BLKSIZE=0,AVGREC=K
   IEFC653I SUBSTITUTION JCL - DSN=HRD.LA.RAWCHKS,DISP=(,PASS)
   BLKSIZE=0
 9 //STEP1.SUMRY  DD  COPIES=3
   X/SUMRY    DD   SYSOUT=&SUMRY
   IEFC653I SUBSTITUTION JCL - SYSOUT=*
10 XXSTEP2   EXEC  PGM=PRCHKS,PARM=&LOCATION
   IEFC653I SUBSTITUTION JCL - PGM=PRCHKS,PARM=('LOS ANGELES')
11 XXINRAW    DD   DSN=HRD.&LOC..RAWCHKS,
   XX    DISP=(OLD,DELETE)
   IEFC653I SUBSTITUTION JCL - DSN=HRD.LA.RAWCHKS,DISP=(OLD,DE
12 XXCHECKS   DD   SYSOUT=V
```

♦ **Notice the IEFC653I messages showing you the result of symbolic substitution**

✗ Wider than the screen sometimes, as in statements 8 and 11 above

Substitution Of Symbolic Parameters

❒ Given a cataloged procedure invoked with the following symbolic parameter values:

```
LIBNAME=LIB

QUALIF=SUIT

LVL2=RARY.TESTED

VAR1=ED
```

❒ Predict what the following DSNs would result in:

```
&LIBNAME&LVL2   —>


&QUALIF.OR.&QUALIF&VAR1   —>


&LIBNAME.EL..&QUALIF   —>


CONT&LVL2..&QUALIF   —>


CONT.&LVL2..&QUALIF&VAR1   —>
```

❒ Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Substitution Of Symbolic Parameters

- ❒ Given a cataloged procedure invoked with the following symbolic parameter values:

  LIBNAME=LIB

  QUALIF=SUIT

  LVL2=RARY.TESTED

  VAR1=ED

- ❒ Predict what the following DSNs would result in:

  &LIBNAME&LVL2 —> `LIBRARY.TESTED`

  &QUALIF.OR.&QUALIF&VAR1 —>

  &LIBNAME.EL..&QUALIF —>

  CONT&LVL2..&QUALIF —>

  CONT.&LVL2..&QUALIF&VAR1 —>

- ❒ Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Substitution Of Symbolic Parameters

- ❐ Given a cataloged procedure invoked with the following symbolic parameter values:

  LIBNAME=LIB

  QUALIF=SUIT

  LVL2=RARY.TESTED

  VAR1=ED

- ❐ Predict what the following DSNs would result in:

  &LIBNAME&LVL2 —> LIBRARY.TESTED

  &QUALIF.OR.&QUALIF&VAR1 —> SUITOR.SUITED

  &LIBNAME.EL..&QUALIF —>

  CONT&LVL2..&QUALIF —>

  CONT.&LVL2..&QUALIF&VAR1 —>

- ❐ Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Substitution Of Symbolic Parameters

- ❒ Given a cataloged procedure invoked with the following symbolic parameter values:

  LIBNAME=LIB

  QUALIF=SUIT

  LVL2=RARY.TESTED

  VAR1=ED

- ❒ Predict what the following DSNs would result in:

  &LIBNAME&LVL2 —> `LIBRARY.TESTED`

  &QUALIF.OR.&QUALIF&VAR1 —> `SUITOR.SUITED`

  &LIBNAME.EL..&QUALIF —> `LIBEL..SUIT  <-- A JCL error`

  CONT&LVL2..&QUALIF —>

  CONT.&LVL2..&QUALIF&VAR1 —>

- ❒ Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Substitution Of Symbolic Parameters

❒ Given a cataloged procedure invoked with the following symbolic parameter values:

LIBNAME=LIB

QUALIF=SUIT

LVL2=RARY.TESTED

VAR1=ED

❒ Predict what the following DSNs would result in:

&LIBNAME&LVL2 —> LIBRARY.TESTED

&QUALIF.OR.&QUALIF&VAR1 —> SUITOR.SUITED

&LIBNAME.EL..&QUALIF —> LIBEL..SUIT

CONT&LVL2..&QUALIF —>

CONT.&LVL2..&QUALIF&VAR1 —>

❒ Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Substitution Of Symbolic Parameters

- ❐ **Given a cataloged procedure invoked with the following symbolic parameter values:**

  LIBNAME=LIB

  QUALIF=SUIT

  LVL2=RARY.TESTED

  VAR1=ED

- ❐ **Predict what the following DSNs would result in:**

  &LIBNAME&LVL2 —> `LIBRARY.TESTED`

  &QUALIF.OR.&QUALIF&VAR1 —> `SUITOR.SUITED`

  &LIBNAME.EL..&QUALIF —> `LIBEL..SUIT`

  CONT&LVL2..&QUALIF —> `CONTRARY.TESTED.SUIT`

  CONT.&LVL2..&QUALIF&VAR1 —>

- ❐ **Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules**

Substitution Of Symbolic Parameters

- Given a cataloged procedure invoked with the following symbolic parameter values:

  LIBNAME=LIB

  QUALIF=SUIT

  LVL2=RARY.TESTED

  VAR1=ED

- Predict what the following DSNs would result in:

  &LIBNAME&LVL2 —> `LIBRARY.TESTED`

  &QUALIF.OR.&QUALIF&VAR1 —> `SUITOR.SUITED`

  &LIBNAME.EL..&QUALIF —> `LIBEL..SUIT`

  CONT&LVL2..&QUALIF —> `CONTRARY.TESTED.SUIT`

  CONT.&LVL2..&QUALIF&VAR1 —> `CONT.RARY.TESTED.SUITED`

- Note that most of these examples are not typical, but we wanted to demonstrate the syntax rules

Symbolic Parameter Names

❒ **You may use any word you like for a symbolic parameter name except for the names of parameters on the EXEC statement**

- ♦ **So, you can't use PGM, TIME, REGION, MEMLIMIT, or PARM as symblic parameter names**

  - ✗ Instead of REGION=&REGION, code, say, REGION=&SIZE

❒ **You may use a symbolic parameter for the program name, for example: PGM=&PRG**

❒ **You may use names of parameters on other JCL statements, or make up your own:**

DISP=&DISP   or   DISP=&DISPOSE

Supplying Default Values: The PROC Statement

❒ The 'PROC' JCL statement allows you to specify defaults for symbolic parameters: then you just specify the particular values you want to override

```
//RRLD23  PROC  PRI=1000,SEC=1000,SUMRY=E
//*   RRLD23 - EDIT EMPLOYEE TIME SLIPS, AND
//*   PRODUCE WORK SUMMARIES AND PRINT CHECKS
//STEP1   EXEC  PGM=IRLD11,PARM=&LOCATION
//TIMESLIP DD   DSN=HRD.&LOC..TIMESLPS,DISP=SHR
//EMPFILE  DD   DSN=HRD.&LOC..EMPLYS,DISP=OLD
//BENTABLE DD   DSN=HRD.CORP.BENEFITS,DISP=SHR
//         DD   DSN=HRD.&LOC..BENEFITS,DISP=SHR
//RAWCHKS  DD   DSN=HRD.&LOC..RAWCHKS,DISP=(,PASS),
//   SPACE=(120,(&PRI,&SEC)),BLKSIZE=0,AVGREC=K
//SUMRY    DD   SYSOUT=&SUMRY
//STEP2   EXEC  PGM=PRCHECKS,PARM=&LOCATION
//INRAW    DD   DSN=HRD.&LOC..RAWCHKS,
//    DISP=(OLD,DELETE)
//CHECKS   DD   SYSOUT=V
```

The PROC Statement, continued

❐ **Then invoke this procedure:**

```
//ANYOLJOB  JOB  (123456,000),'PP',CLASS=R,
//    MSGCLASS=A,NOTIFY=LOGGER
//RUNIT    EXEC  RRLD23,LOCATION=('LOS ANGELES'),
//    LOC=LA
```

- ♦ **And the procedure is invoked with all its symbolics assigned their default values, except for 'LOCATION' (which is assigned 'LOS ANGELES' this time), and 'LOC' (which is assigned 'LA')**

❐ **Again, you can combine overrides and inserts with symbolic parameter substitution**

SYSUID

❒ **The reserved symbolic parameter &SYSUID resolves to the userid from which the job was submitted**

- ♦ **Could use as a high-level qualifier for a data set name, for example**

```
//STEPLIB  DD  DSN=&SYSUID..TRAIN.LOAD,DISP=SHR
```

- ♦ **Do not use this in the name field of the JOB statement**

    - ✗ But you <u>may</u> specify NOTIFY=&SYSUID on the JOB statement

- ♦ **Do not use this in the accounting information or programmer name fields in the JOB statement**

- ♦ **Do not use this on the XMIT statement, or any JES2 or JES3 control statements**

    - ✗ Probably not a problem, since we haven't talked about these statements

Coding Procedures - A Strategy

❒ **When you need to create a procedure, consider this approach:**

- ♦ **Code the JCL as it will appear in a typical invocation, with all EXEC, DD, OUTPUT, and other JCL statements**

- ♦ **Code a PROC statement at the front**

- ♦ **Remove DD statements that describe data that is usually sysin type data**

- ♦ **Select variables for parameters that are likely to change most times the procedure is invoked**

  - ✗ Consider: PARM on EXEC statements, DSN and SPACE parameters on non-SYSOUT DD statements, SYSOUT class, COPIES, and DEST on SYSOUT DD statements; possibly also DCB-related parameters

- ♦ **Document (comment) each variable at the front of the proc as you place variables in the body of the proc**

  - ✗ Choose variable names that are descriptive / helpful

  - ✗ Note usage, intended values, default value (if any), etc.

- ♦ **If a parameter is frequently the same, code the most common value as a default in the PROC statement**

- ♦ **Test, test, test**

This page intentionally left almost blank.

Computer Exercise: A Procedure With Symbolic Parameters

Modify your COPYNPRT procedure so that it has this structure:

PROC

PROC statement with default values for: primary and secondary
space requests on the OUTDD statement in step one, and
the member name for the SYSIN statement in step two

EXEC statement to run NEWF2F

STEPLIB DD statement; point to ________.TRAIN.LOADLIB (SHR!)

INDD DD statement: DISP=SHR; DSN a symbolic; no default

OUTDD DD statement: DISP=(,PASS); DSN a temporary data set name;
UNIT is SYSDA; SPACE units in terms of 100-byte records;
primary and secondary units as symbolics, with default values of
100 and 50, respectively; AVGREC as U

EXEC statement to run SORT

SORTIN DD statement; point to OUTDD data set from above step,
DISP is old, delete

SORTOUT DD statement; point to printer

SYSIN DD statement; DISP=SHR, DSN= _______.TRAIN.LIBRARY with
a symbolic for a member name (default = CNTRL2)

SYSOUT DD statement; point to printer

****** more on next page ******

Computer Exercise: A Procedure With Symbolic Parameters, 2

Create JCLEX11, based on JCLEX10, and make sure JCLEX11 has:

JOB to invoke the proc

A JOB statement

NOTE: make sure your JOB statement has a NOTIFY that specifies &SYSUID

A JCLLIB statement referencing your <userid>.TR.CNTL

An EXEC statement to run your proc, specifying the following symbolic parameter values:

DSN for INDD: ______.TRAIN.INPUTX

CNTRL3 for member name on the SYSIN DD statement

An EXEC statement to run your proc, specifying the following symbolic parameter values:

DSN for INDD: ______.TRAIN.LIBRARY(INPUTC)

75 for primary space on OUTDD, and 40 for secondary

In other words, you're testing your proc twice in one run.

Run the job.

Section Preview

❒ **JCL SETs, INCLUDEs and Nested Procedures**

- **The SET statement**
- **The INCLUDE statement**
- **Nested Procedures**
- **Using Nested Procedures and INCLUDES (Optional Machine Exercise)**

This page intentionally left almost blank.

Symbolic Parameters in Open JCL

❒ **Symbolic parameters may appear in "open JCL" as well as in cataloged or in-stream procedures**

- **A value is assigned to a symbolic parameter in one of three ways**

  ✗ On the PROC statement of a cataloged or instream procedure (this provides a default value)

```
//PR342    PROC    DSP=SHR,SSP=5
```

  ✗ On the EXEC statement that invokes a procedure, for example

```
//DOIT    EXEC   PR342,DSP=OLD,PSP=10
```

  ✗ On the SET statement (discussed next)

```
//PRE     SET    SUN=TRK,NUM=32
```

Order of Precedence

- **EXEC statement values when invoking a procedure override default values on the PROC statement**

- **SET values only take place in a procedure if there has been no value assigned through the PROC or EXEC statements**

SET Statement

❒ **The formal syntax of SET is**

//[*name*] **SET** *symbolic*=*value*[,*symbolic*=*value*[,...]]

<u>**Examples**</u>

```
//THEJOB   JOB   ----------
//PRESET   SET   TIMES=1,LVL1=PAYROLL
//STEP1    EXEC  PROC=ONEPROC
. . .
//FIRSTSET SET   TIMES=2
//STEP2    EXEC  PROC=CROPPROC,LVL1=TPAYROLL
//          SET  TIMES=3,SZ=1024K
//LITTLE   EXEC  PGM=TOOTINY,REGION=&SZ,PARM=&TIMES
//TOODD1    DD   DSN=&LVL1..MASTER,DISP=SHR
//SYSOUT    DD   SYSOUT=*
//TOTO     EXEC  PGM=TOOBIG,REGION=&SZ,PARM=&TIMES
//ANYDD     DD   -------------
//ANYDOODO  DD   -------------
```

- **SET values remain in effect until a subsequent SET statement changes them**
- **SET values do not override explicit assignments in PROC or EXEC statements**
- **SET values <u>do</u> apply in a procedure if the symbolic has not been assigned or nullified in a PROC or EXEC statement**
- **SET values inside a procedure do not have an effect outside their procedure**

SET Statement, 2

More examples

```
//THEJOB   JOB   ----------
//PRESET   SET   TIMES=1,LVL1=PAYROLL
//STEP1    EXEC  PROC=ONEPROC
. . .
//FIRSTSET SET   TIMES=2
//STEP2    EXEC  PROC=CROPPROC,LVL1=TPAYROLL
//          IF   STEP2.STEP23.RC = 0  THEN
//         SET   TIMES=3,SZ=1024K
//LITTLE   EXEC  PGM=TOOTINY,REGION=&SZ,PARM=&TIMES
//TOODD1    DD   DSN=&LVL1..MASTER,DISP=SHR
//SYSOUT    DD   SYSOUT=*

//          ELSE
//TOTO     EXEC  PGM=TOOBIG,REGION=&SZ,PARM=&TIMES
//ANYDD     DD   -------------
//ANYDOODO  DD   -------------
//          ENDIF
```

- **SET statements are processed regardless of IF / THEN, ELSE, ENDIF statements**
  - ✗ SET is not conditional

- **SET statement(s) must appear after a JOB statement and before the end of the job**

- **Place a SET statement in the JCL prior to the first use of any symbolic parameter(s) on the SET statement**

INCLUDE

❒ **You can create <u>INCLUDE groups</u> of pre-coded JCL and store them in a library**

❒ **Then bring in such groups using the INCLUDE statement**

<u>Syntax</u>

//[name]   INCLUDE   MEMBER=*name*

<u>Statements that can be coded in INCLUDE groups</u>

✗ EXEC

✗ DD

✗ OUTPUT

✗ IF / THEN, ELSE, ENDIF

✗ Comment

✗ INCLUDE (nest up to 15 levels)

✗ SET

<u>Statements that may not be included in INCLUDE groups</u>

✗ JOB

✗ PROC, PEND, JCLLIB

✗ JES2, JES3 statements

✗ DD   * or DD   DATA   (SYSIN data; no delimiters, either)

Special note: from z/OS 1.13, JES2 systems <u>can</u> have SYSIN data inside an INCLUDE group

INCLUDE, 2

❐ The named member is brought in at the location of the INCLUDE statement, following the INCLUDE statement itself

- ♦ Symbolic parameters are assigned their current values

How is an INCLUDE group different from a procedure?

- ♦ INCLUDE groups do not need to contain any EXEC statements, procedures must contain at least one
- ♦ INCLUDE groups may not contain procedure definitions, procedures are their own definitions
- ♦ You cannot override statements in INCLUDE groups, you can override and insert statements in procedures (*e.g.*: //stepname.ddname DD ...)
- ♦ SET statements in an INCLUDE apply to subsequent JCL

Examples of INCLUDE

```
//SETUP  INCLUDE  MEMBER=START12


//FIXIT  INCLUDE  MEMBER=RECLAIM
```

INCLUDE, 3

❒ Suppose the include group START12 contained the following

```
//*   INCLUDE_MEMBER_NAME IS: START12
//DDN1       DD   DISP=(,CATLG),DSN=&NEW1
//DDN2       DD   DISP=(,CATLG),DSN=&NEW2
//DDO1       DD   DISP=(OLD,DELETE),DSN=&OLD1
//DDO2       DD   DISP=(OLD,DELETE),DSN=&OLD2
```

❒ If you submitted the following job

```
//TRIED2    JOB  -------------
//         SET  NEW1='ISD4.BADTRANS',NEW2=NULLFILE,
//    OLD1='ISD4.GOODTR',OLD2=NULLFILE
//PREALLOC  EXEC  PGM=IEFBR14
//          INCLUDE  MEMBER=START12
//. . .
```

♦ The resulting JCL would appear on your JCL listing as:

```
//TRIED2    JOB  -------------
//         SET  NEW1='ISD4.BADTRANS',NEW2=NULLFILE,
//    OLD1='ISD4.GOODTR',OLD2=NULLFILE
//PREALLOC  EXEC  PGM=IEFBR14
//          INCLUDE  MEMBER=START12
XX*   INCLUDE_MEMBER_NAME IS: START12
XXDDN1       DD   DISP=(,CATLG),DSN=ISD4.BADTRANS
XXDDN2       DD   DISP=(,CATLG),DSN=NULLFILE
XXDDO1       DD   DISP=(OLD,DELETE),DSN=ISD4.GOODTR
XXDDO2       DD   DISP=(OLD,DELETE),DSN=NULLFILE
//. . .
```

Nested Procedures

❒ A cataloged or instream procedure may in turn invoke another procedure by including an EXEC statement that names the procedure:

```
//DSTP    EXEC  PROC=PYRPT
```

Or

```
//DSTP    EXEC  PYRPT
```

❒ Recursion is not allowed

- That is, a cataloged procedure may not invoke itself
- Nor may a cataloged procedure invoke a procedure that invoked it

Nested Procedures, 2

❒ **For example, suppose you have these members stored in the private proclib DEPT22.PROCS:**

<u>OUTIE</u>

```
//STP1  EXEC  PGM=ABC,PARM=MAYBE
//COIN   DD   DISP=SHR,DSN=ANY.MASTER
//COUT   DD   DISP=(,CATLG),DSN=&SYSUID..KEEPER
//STP2  EXEC  INNIE
//STP3  EXEC  PGM=NEWF
//    INCLUDE  MEMBER=NEWSDDS
```

<u>INNIE</u>

```
//OFF1  EXEC  PGM=IRD55
//OFFDD1 DD   DISP=SHR,DSN=&SYSUID..KEEPER
//SUMMRY DD   SYSOUT=*
//       IF   OFF1.RC=0  THEN
//CLEAN EXEC  PGM=IEFBR14
//ONLYDD DD   DSN=&SYSUID..KEEPER,DISP=(OLD,DELETE)
//      ENDIF
```

<u>NEWSDDS</u>

```
//*   INCLUDE MEMBER NAME IS NEWSDDS
//XXDD   DD  DISP=SHR,DSN=AARDVARK.PAYS
//YYDD   DD  SYSOUT=*
//ZZDD   DD  DSN=&SYSUID..TOP,DISP=OLD
```

Nested Procedures, 3

- ❒ **Now, if your TSO-id is U0034, then the following JCL:**

```
//BIGJOB   JOB   ----------------
//MYJCL    JCLLIB  ORDER=(DEPT22.PROCS)
//DOIT     EXEC  OUTIE
```

  - ♦ **Will produce this result after processing by the converter:**

```
//BIGJOB   JOB   ----------------
//MYJCL    JCLLIB  ORDER=(DEPT22.PROCS)
//DOIT     EXEC  OUTIE
XXSTP1  EXEC  PGM=ABC,PARM=MAYBE
XXCOIN   DD   DISP=SHR,DSN=ANY.MASTER
XXCOUT   DD   DISP=(,CATLG),DSN=U0034.KEEPER
XXSTP2  EXEC  INNIE
XXOFF1  EXEC  PGM=IRD55
XXOFFDD1 DD   DISP=SHR,DSN=U0034.KEEPER
XXSUMMRY DD   SYSOUT=*
XX       IF   OFF1.RC=0  THEN
XXCLEAN EXEC  PGM=IEFBR14
XXONLYDD DD   DSN=U0034.KEEPER,DISP=(OLD,DELETE)
XX      ENDIF
XXSTP3   EXEC  PGM=NEWF
XX     INCLUDE  MEMBER=NEWSDDS
***   INCLUDE MEMBER NAME IS NEWSDDS
XXXXDD   DD  DISP=SHR,DSN=AARDVARK.PAYS
XXYYDD   DD  SYSOUT=*
XXZZDD   DD  DSN=U0034.TOP,DISP=OLD
```

Nested Procedures, 4

❒ **Nesting may be up to 15 levels deep, with a few restrictions:**

- **Modifying, overriding, inserting statements may only be done one level deep**

  ✗ For example, the following is allowed

```
//stepname.ddname  DD  ...
```

  ✗ But the following is not allowed

```
//procstepname.stepname.ddname  DD  ...
```

- **Cannot make backwards references to DD statements in nested procedures**

  ✗ For example, the following is not allowed

```
//RINP  DD  DSN=*.INVK.RINK.STEP11.TMSLP,DISP=SHR
```

- **Cannot modify DD statements that are modifying nested DD statements already**

<u>Computer Exercise</u>: Using Nested Procedures and INCLUDES
(Optional, time permitting)

Here is an opportunity to explore some of the new JCL facilities.

1. In your JCL data set, create a member called FILES, consisting of these lines:

```
//INDD    DD  DISP=SHR,DSN=&SYSUID..&MID..&LOW(INPUT2)
//OUTDD   DD  SYSOUT=*
```

2. In your JCL data set, code a member called CCODE, consisting of these lines

```
//CCODE     PROC
//RUNPR     EXEC     PGM=NEWF2F,PARM=&TEXT
//          INCLUDE  MEMBER=FILES
```

3. In your JCL data set, create member JCLEX12, with this JCL
   (add any necessary JES control statements):

```
//jobname   JOB  (acctng info),pgmr-name,CLASS=_,
//   MSGCLASS=_,NOTIFY=&SYSUID
//JOBLIB     DD      DISP=SHR,DSN=_______.TRAIN.LOADLIB
//           SET     MID=TR,TEXT=FIVE,LOW=CNTL
//PLIBS      JCLLIB  ORDER=(&SYSUID..&MID..&LOW)
//FIRST      EXEC    CCODE
//           IF      FIRST.RUNPR.RC=4  THEN
//SECOND1    EXEC    CCODE,TEXT=SECOND1
//           ELSE
//SECOND2    EXEC    CCODE,TEXT=SECOND2
//           ENDIF
```

4. Run this job. Which step was executed, SECOND1 or SECOND2?

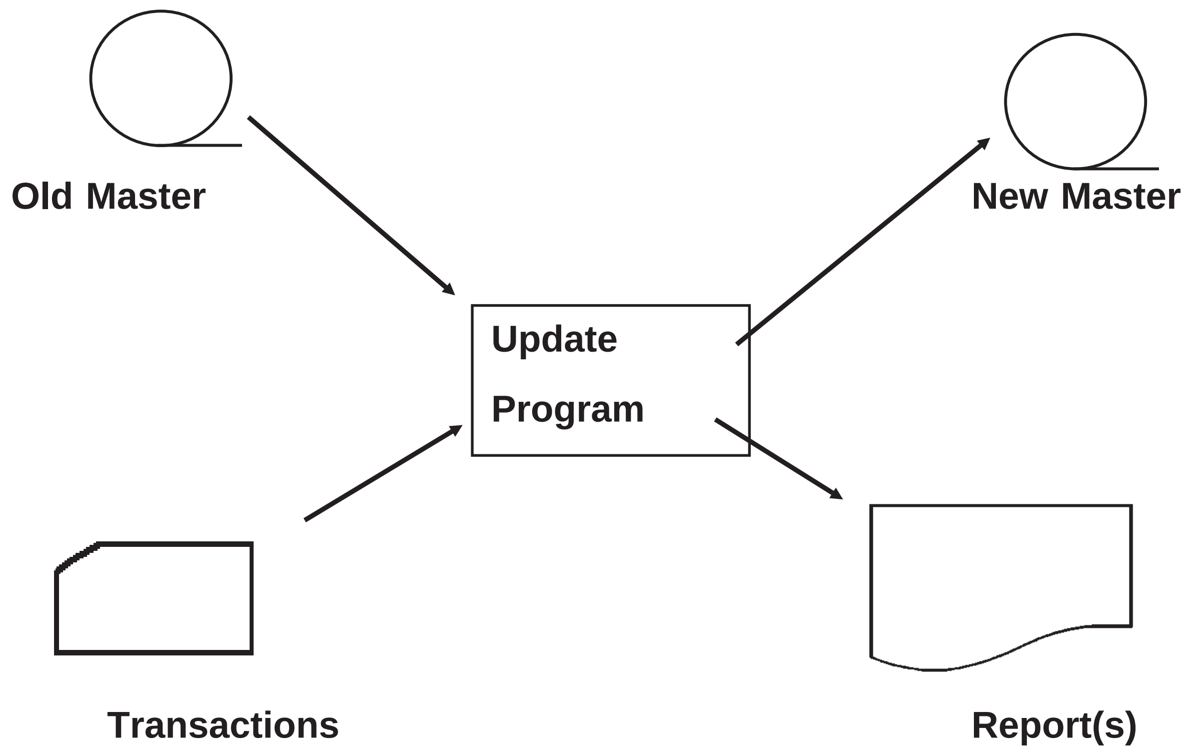
This page intentionally left almost blank.

Section Preview

❐ **Additional Data Set Handling Techniques**

- ♦ **GDGs - Generation Data Groups**
- ♦ **PDSEs - Partitioned Data Set, Extended**

The Classical Data Processing Application



Consider the JCL for running this step today: then, what about tomorrow?

```
//RUNIT    EXEC  PGM=MASTUPDT
//OLDMAST   DD   DSN=OLD.CUSTOMER.HISTORY,DISP=SHR
//NEWMAST   DD   DSN=NEW.CUSTOMER.HISTORY,DISP=(,CATLG),UNIT=TAPE
 ...
```

Generation Data Group (GDG)

- ❒ **A collection of successive, historically related, cataloged data sets**
  - ♦ **Each data set in the collection is called a generation data set**

- ❒ **Define the group using the AMS command DEFINE GDG:**

```
DEFINE GDG      (NAME(CUSTOMER.HISTORY) -
                LIMIT(13) -
                NOEMPTY -
                SCRATCH)
```

<u>Where...</u>

- ♦ **The LIMIT value specifies how many generations the system should keep track of (1-255)**
- ♦ **NOEMPTY says when the GDG index is full, add the new entry at the top, slide everyone down one slot, and drop the last entry off the index (as opposed to EMPTY which says drop all entries when the index is full); "drop" here means remove from the index and uncatalog**
- ♦ **SCRATCH says when a data set is dropped from the index it should be scratched (deleted from the catalog and the VTOC) (as opposed to NOSCRATCH, which will uncatalog but not delete the data set)**

Using GDGs

❒ **Generation data sets are referenced by coding the group name followed by a <u>relative generation number</u> in parentheses**

CUSTOMER.HISTORY(0)	—	The latest generation of the data set
CUSTOMER.HISTORY(+1)	—	The first new generation of this data set created during this job
CUSTOMER.HISTORY(+2)	—	The second new generation of this data set created during this job
CUSTOMER.HISTORY(-1)	—	The generation created before the latest generation

❒ **Typically, you read in the latest generation and produce a new latest generation, so you would code something like this:**

```
//--------    JOB     --------
//TRANS       EXEC    PGM=MASTUPDT
//OLDMAST     DD   DSN=CUSTOMER.HISTORY(0),DISP=SHR
//NEWMAST     DD   DSN=CUSTOMER.HISTORY(+1),
//         DISP=(,CATLG,DELETE),SPACE=(CYL,(20,3),RLSE)
```

- ♦ **At end of step, the new data set is cataloged**

- ♦ **At end of job, the GDG index is updated**

  - ✗ **If the index is full, either the oldest generation, or all generations (depending on how the GDG was defined, EMPTY or NOEMPTY) are uncataloged and (if SCRATCH was specified in the DEFINE) scratched from the DASD volume**

GDGs: Other Notes

❒ **Generation data sets are actually stored in the catalog under names of the form:** *groupname***.G***nnnn***V***mm*

- **Where:**

  ✗ *nnnn* starts out at 0001 and is incremented by one each time a new generation is added, wrapping around to 0001 after 9999

  ✗ *mm* is normally stored as 00

Example: **CUSTOMER.HISTORY.G0123V00**

would be the 123rd generation of the GDG

- **If you need to produce a new version of a particular generation, you must catalog it explicitly with the name** *groupname***.G***nnnn***V01**

❒ **This actual naming information is seldom needed, except in some error situations: most times you can use RESTART and override data set names in the JCL with different relative generation numbers (see the JCL and RESTART manuals)**

❒ **Caution needs to be taken if two shared processors can be creating new generations of the same GDG at the same time: best to schedule work so this can't happen**

PDSE - Partitioned Data Set, Extended

❒ **PDSEs are an improved version of PDSs with the following characteristics**

- ♦ **Up to 123 extents (16 for PDS)**
- ♦ **Directory expands and contracts as necessary (PDS directory is fixed until data set reallocated)**
- ♦ **Space left by deleting members is automatically reused (PDSs must be compressed)**
- ♦ **PDSE directory is indexed, PDS directory is not**
- ♦ **May share at the data set level or the member level**
- ♦ **May create multiple members simultaneously**
- ♦ **Non-VSAM (both PDSEs and PDSs)**
- ♦ **PDSEs may not be used to hold load modules but they may contain program objects, a new alternative to load modules**

Creating PDSEs

- **Indicate a new data set is a library (PDS or PDSE) by coding**

  - **DSORG=PO or SPACE=(*unit*,(*pri*,*sec*,*dir*)) on DD statement**

- **The JCL DD statement parameter**

  DSNTYPE={PDS|LIBRARY}

  - **Defines a file as being a PDS (DSNTYPE=PDS) or PDSE (DSNTYPE=LIBRARY)**

- **The default for DSNTYPE is installation-chosen**

- **A DSNTYPE value may also be specified in a DATACLAS or copied from a data set using the LIKE parameter**

Working With PDSEs

❒ **Using PDSEs appears to the user to be the same as using PDSs, except for the improvements mentioned earlier**

❒ **That is, the interfaces are the same for:**

- **JCL**
- **Programming languages**
- **TSO commands**
- **REXX EXECs and CLISTs**
- **ISPF/PDF EDIT and BROWSE**
- **Utilities (*e.g.,* IEBCOPY, IEBGENER, *etc*.)**

❒ **The IEBCOPY utility may be used to copy from a PDS to a PDSE, and vice versa**

Section Preview

❐ **Sources of Information**

- ♦ **IBM Publications**
- ♦ **IBM Web-based Information**
- ♦ **Book Manager**
- ♦ **Quick Reference**
- ♦ **Quick Reference and Job Listings**

IBM Publications

❐ **The IBM publications most useful for help with or information about z/OS JCL and Utilities are these:**

Form Number	Title
z/OS Release 1.x	
SA22-7598	JCL User's Guide
SA22-7597	JCL Reference
SA22-7626	System Codes
SA22-7631- SA22-7639	System Messages
DFSMSdfp for z/OS	
SC26-7401	Checkpoint/Restart
SC26-7414	Utilities
SC26-7326	Access Method Services
DFSORT	
SC33-4035	Application Programming Guide

❐ **These publications are available for a charge in hardcopy and for free on the Internet in PDF and Book Manager formats**

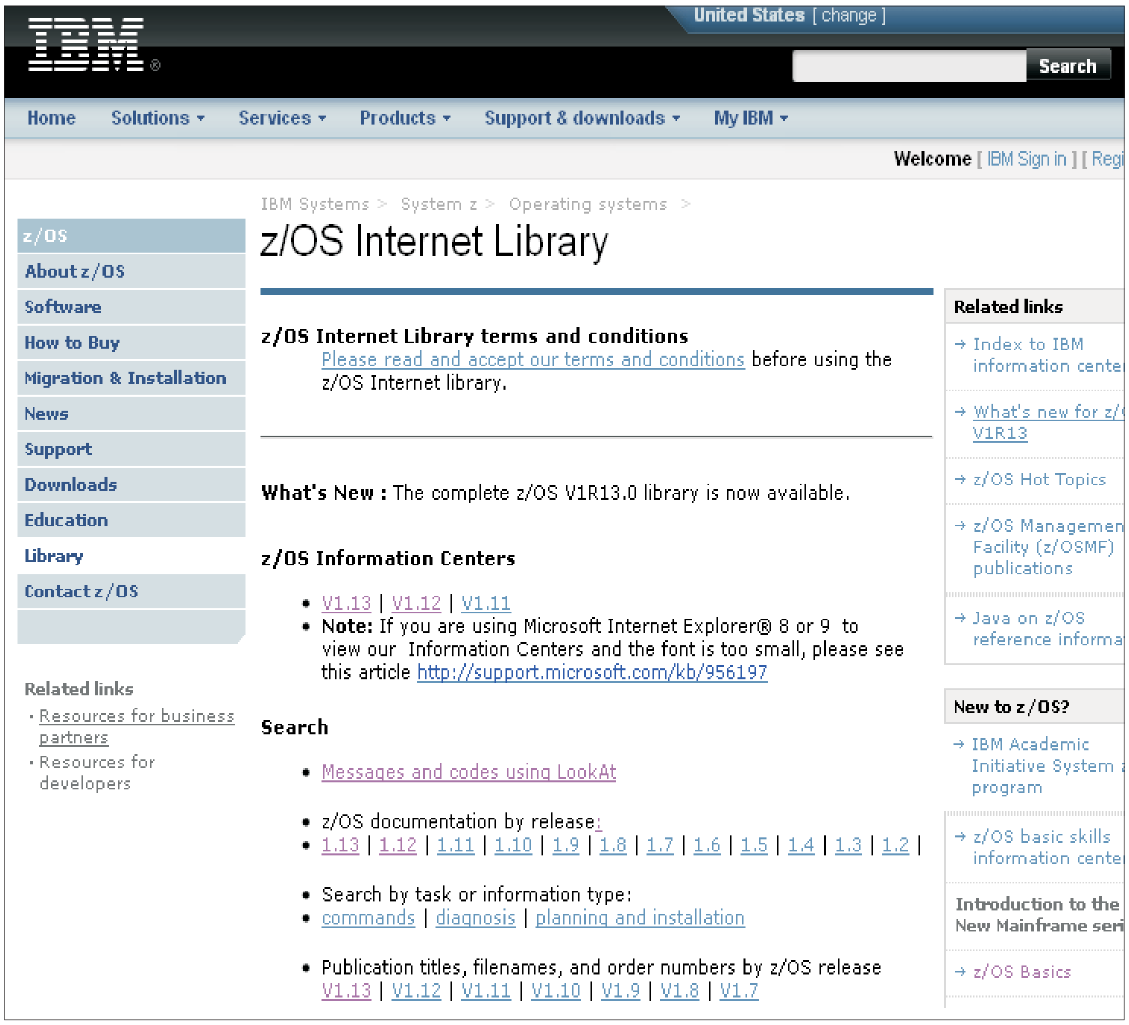
IBM Web-based Information

❒ You can find a wealth of free supporting documentation and help about IBM products on the web

❒ To start, point your browser to:

http://www.ibm.com/systems/z/os/zos/bkserv/

♦ Which will get you to this page (only shown partially here):



IBM Systems > System z > Operating systems >

z/OS Internet Library

z/OS Internet Library terms and conditions
Please read and accept our terms and conditions before using the z/OS Internet library.

What's New : The complete z/OS V1R13.0 library is now available.

z/OS Information Centers

- V1.13 | V1.12 | V1.11
- **Note:** If you are using Microsoft Internet Explorer® 8 or 9 to view our Information Centers and the font is too small, please see this article http://support.microsoft.com/kb/956197

Search

- Messages and codes using LookAt
- z/OS documentation by release:
- 1.13 | 1.12 | 1.11 | 1.10 | 1.9 | 1.8 | 1.7 | 1.6 | 1.5 | 1.4 | 1.3 | 1.2 |
- Search by task or information type:
- commands | diagnosis | planning and installation
- Publication titles, filenames, and order numbers by z/OS release V1.13 | V1.12 | V1.11 | V1.10 | V1.9 | V1.8 | V1.7

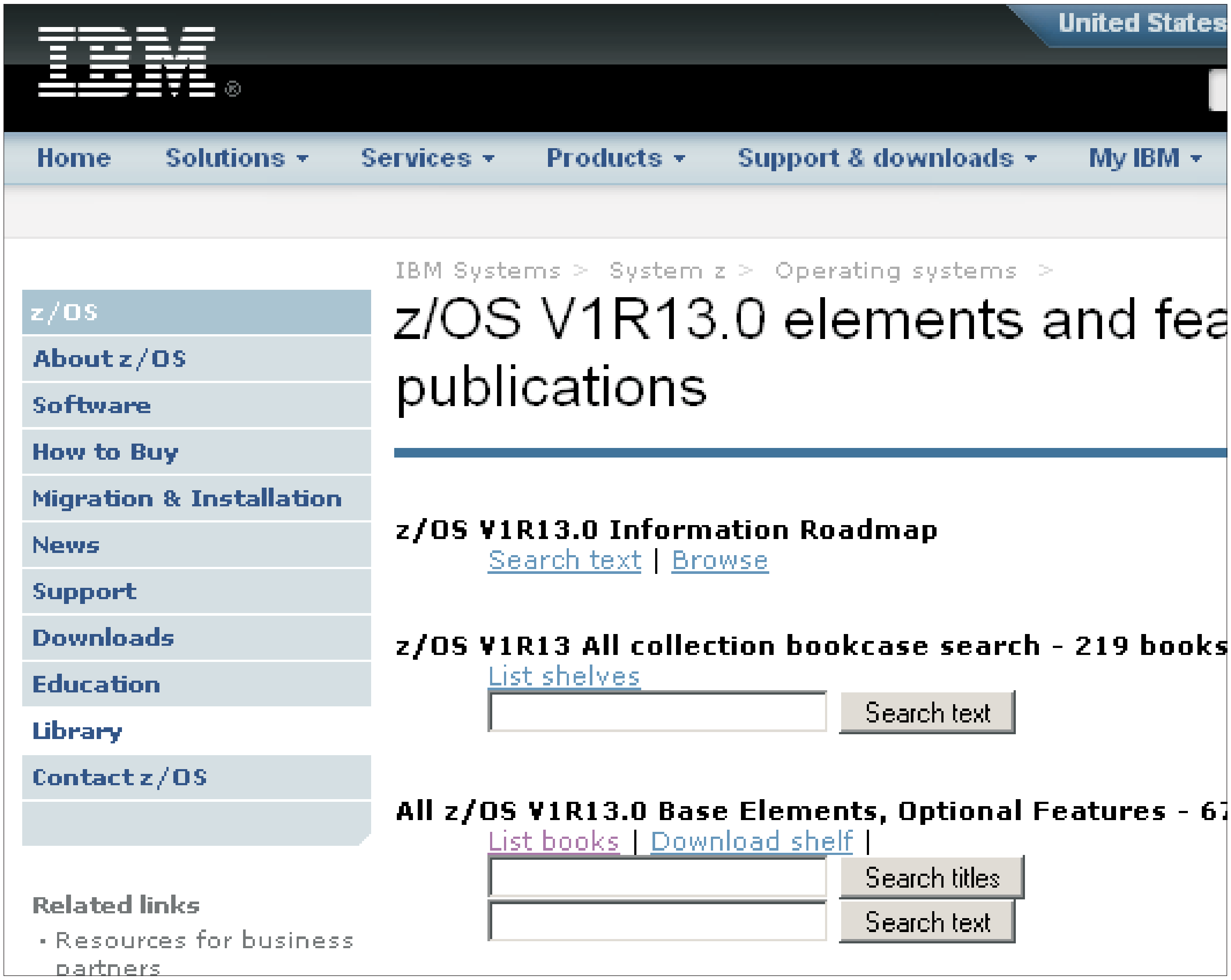
Related links
→ Index to IBM information cente
→ What's new for z/(V1R13
→ z/OS Hot Topics
→ z/OS Managemen Facility (z/OSMF) publications
→ Java on z/OS reference informa

New to z/OS?
→ IBM Academic Initiative System program
→ z/OS basic skills information cente
Introduction to the New Mainframe seri
→ z/OS Basics

♦ Note: the layout changes a bit from release to release

IBM Web-based Information, continued

❒ Under "Publication titles, filenames, ...", click on the version you want (say V1R13) brings you to this page::



z/OS V1R13.0 elements and fea
publications

z/OS V1R13.0 Information Roadmap
Search text | Browse

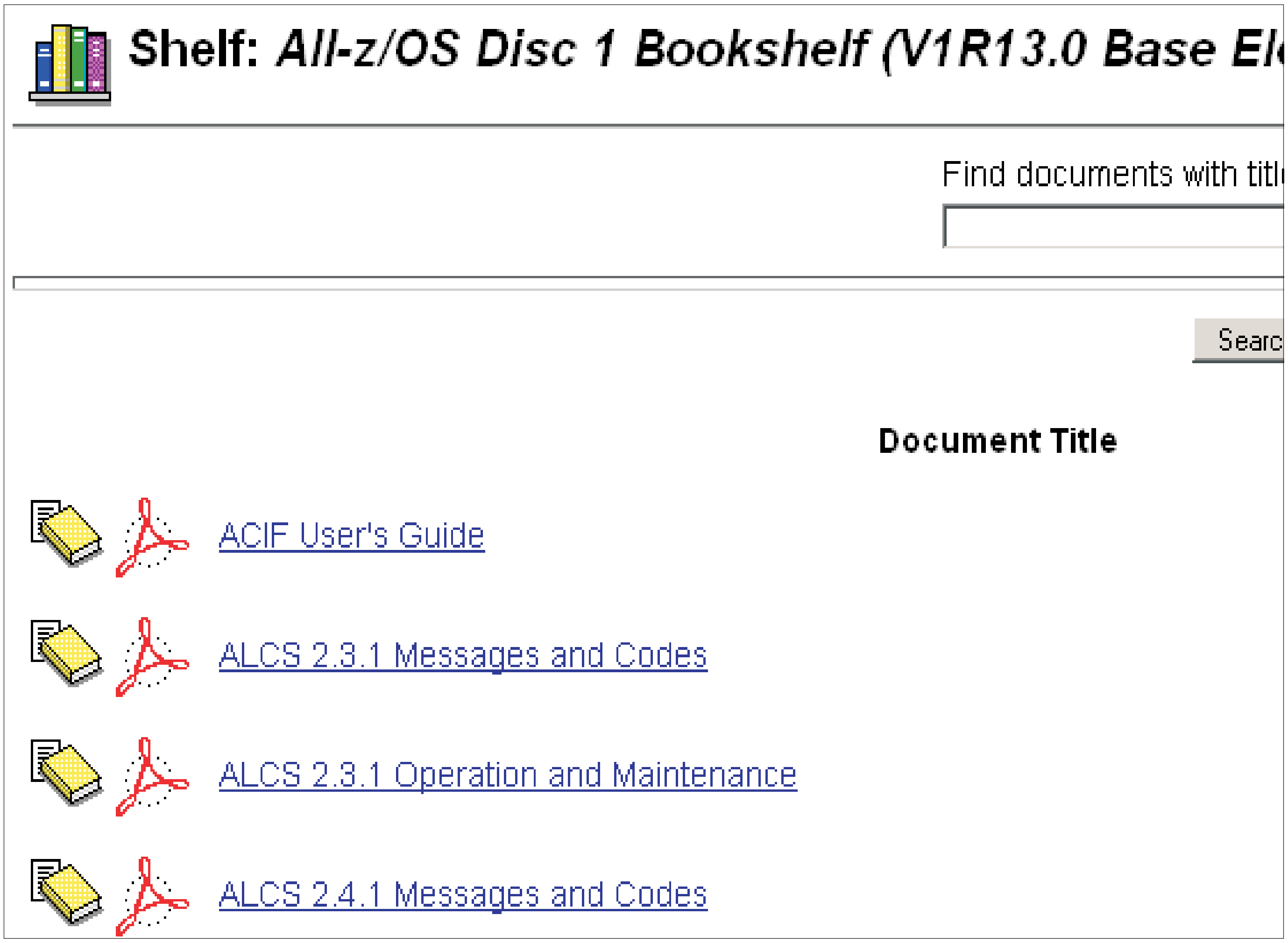
z/OS V1R13 All collection bookcase search - 219 books
List shelves

All z/OS V1R13.0 Base Elements, Optional Features - 67
List books | Download shelf |

♦ And clicking on List books under "All z/OS V1R13 ..." takes you here ...

IBM Web-based Information

❒ **List of books:**



Shelf: ***All-z/OS Disc 1 Bookshelf (V1R13.0 Base El***

Find documents with titl

Searc

Document Title

ACIF User's Guide

ALCS 2.3.1 Messages and Codes

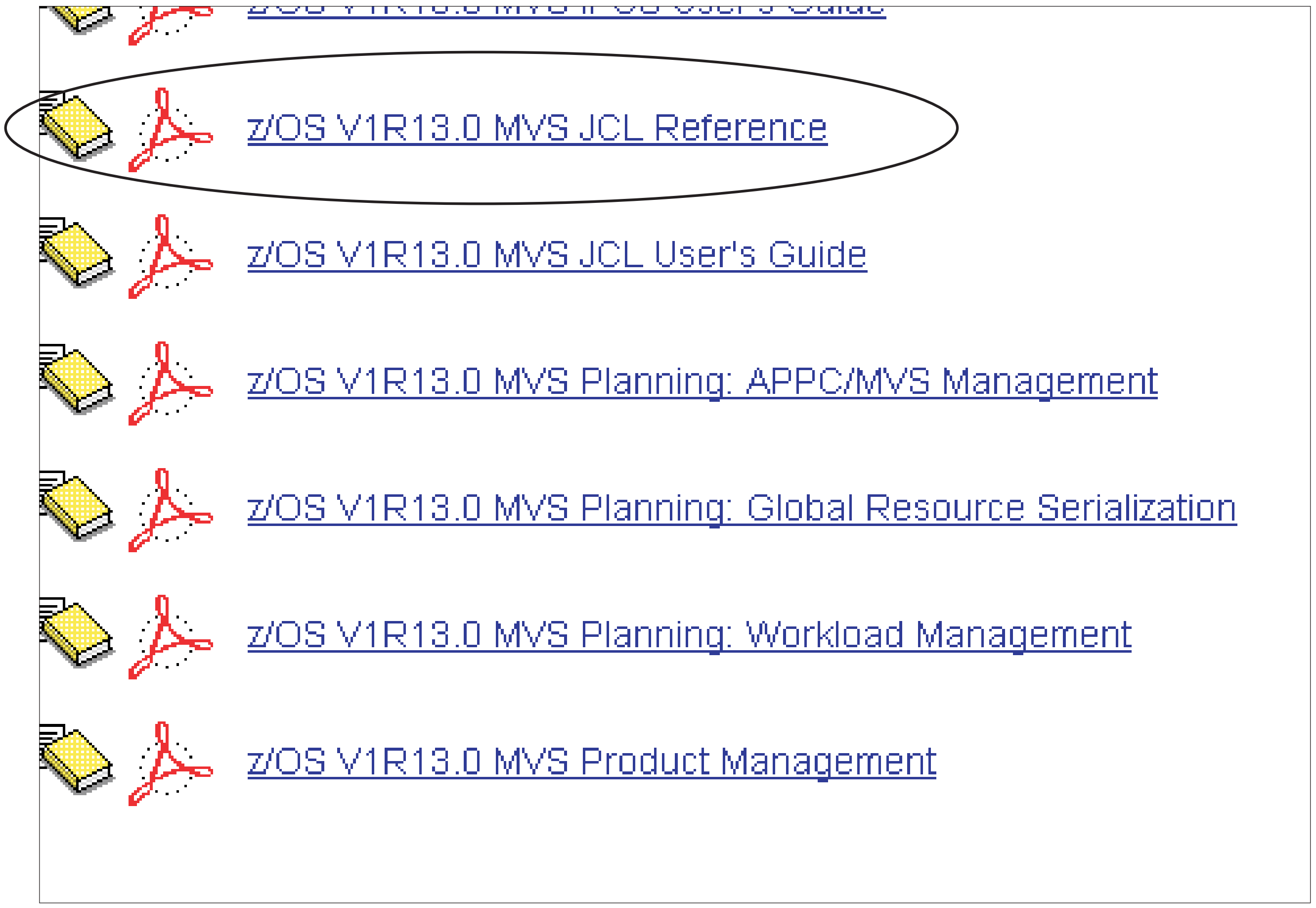
ALCS 2.3.1 Operation and Maintenance

ALCS 2.4.1 Messages and Codes

♦ **Scroll down looking for the book(s) you are after ...**

IBM Web-based Information

❐ **Scroll down and click on the document you are interested in (JCL Reference in the diagram)**



z/OS V1R13.0 MVS JCL Reference

z/OS V1R13.0 MVS JCL User's Guide

z/OS V1R13.0 MVS Planning: APPC/MVS Management

z/OS V1R13.0 MVS Planning: Global Resource Serialization

z/OS V1R13.0 MVS Planning: Workload Management

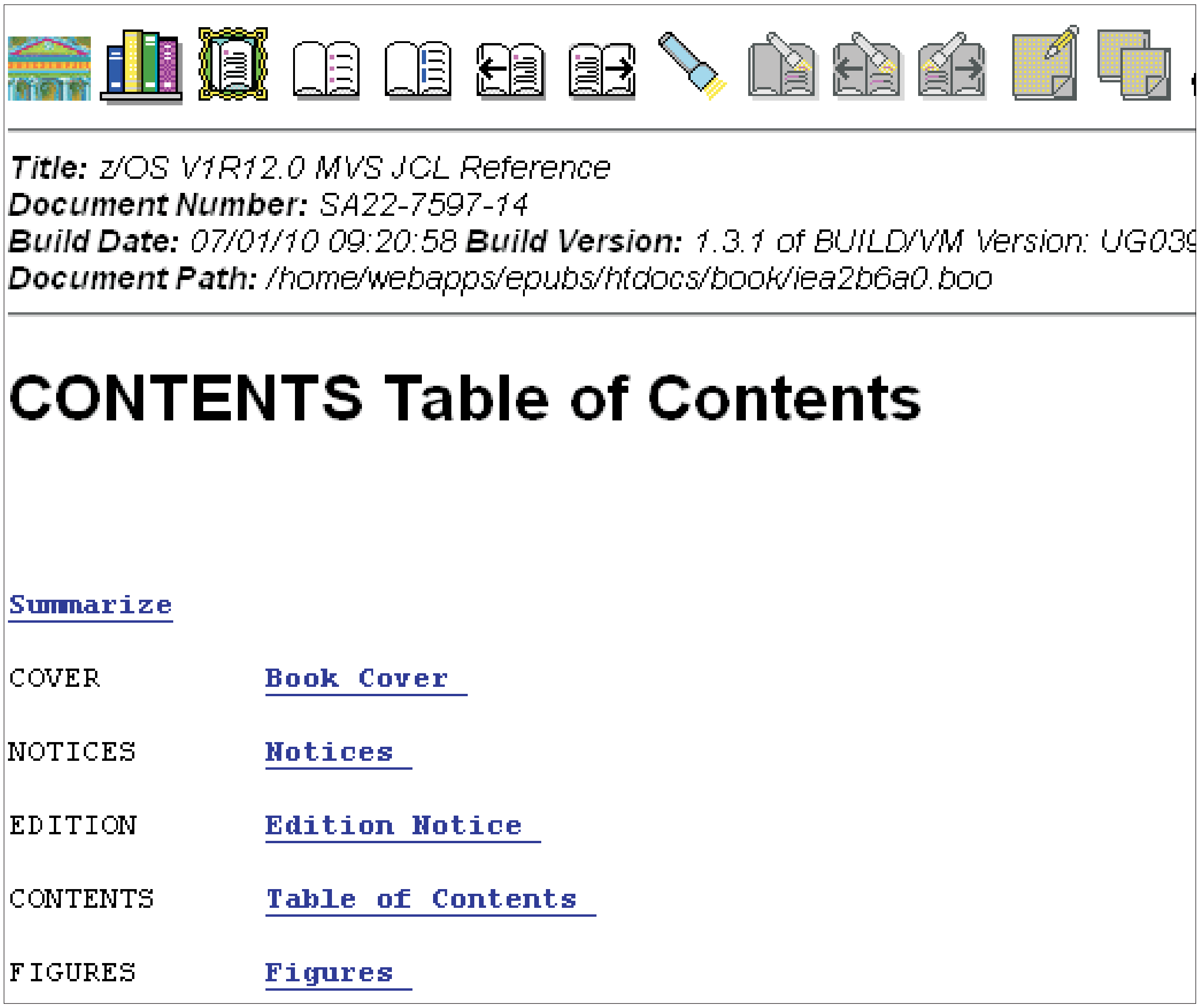
z/OS V1R13.0 MVS Product Management

- **If there is an Adobe Acrobat icon, clicking on the icon loads the book into Adobe Acrobat Reader inside your browser**

- **Clicking on either the Book Manager icon or the link puts you in Book Manager view of the document ...**

IBM Web-based Information, continued

❒ **Book Manager format looks like this:**



Title: *z/OS V1R12.0 MVS JCL Reference*
Document Number: *SA22-7597-14*
Build Date: *07/01/10 09:20:58* **Build Version:** *1.3.1 of BUILD/VM Version: UG039*
Document Path: */home/webapps/epubs/htdocs/book/iea2b6a0.boo*

CONTENTS Table of Contents

Summarize

COVER	Book Cover
NOTICES	Notices
EDITION	Edition Notice
CONTENTS	Table of Contents
FIGURES	Figures

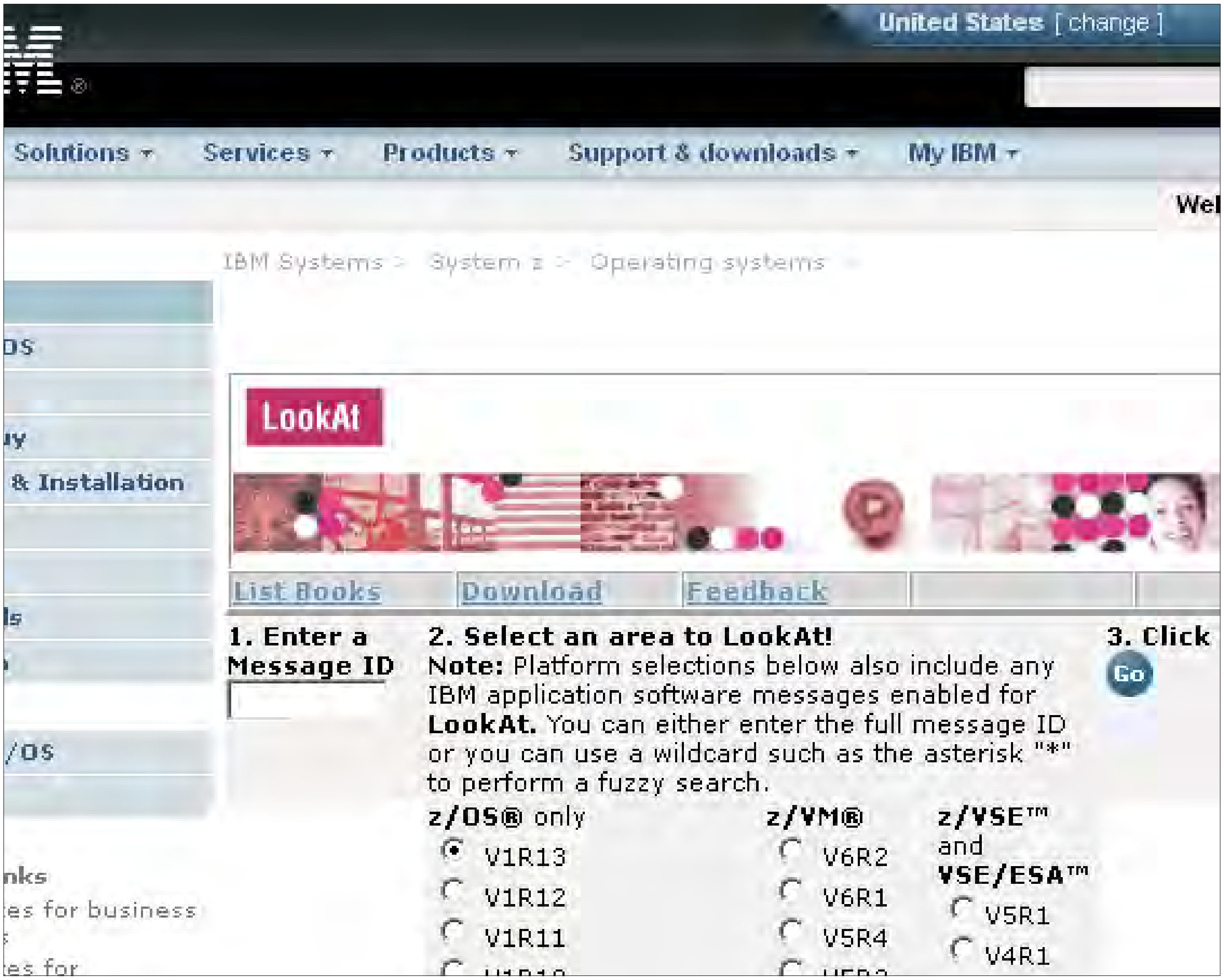
♦ **The icons across the top let you download a publication (in bookmanager format or Adobe Acrobat format, whichever formats are available), search the document, read the document, and so on**

IBM Web-based Information, continued

❒ **If you just need to look up the explanation for a message, you can use the Look-At facility, found at**

http://www.ibm.com/systems/z/os/zos/bkserv/lookat/

- **Which brings you to this page:**



1. Enter a Message ID

2. Select an area to LookAt!
Note: Platform selections below also include any IBM application software messages enabled for **LookAt.** You can either enter the full message ID or you can use a wildcard such as the asterisk "*" to perform a fuzzy search.

z/OS® only
(●) V1R13
( ) V1R12
( ) V1R11

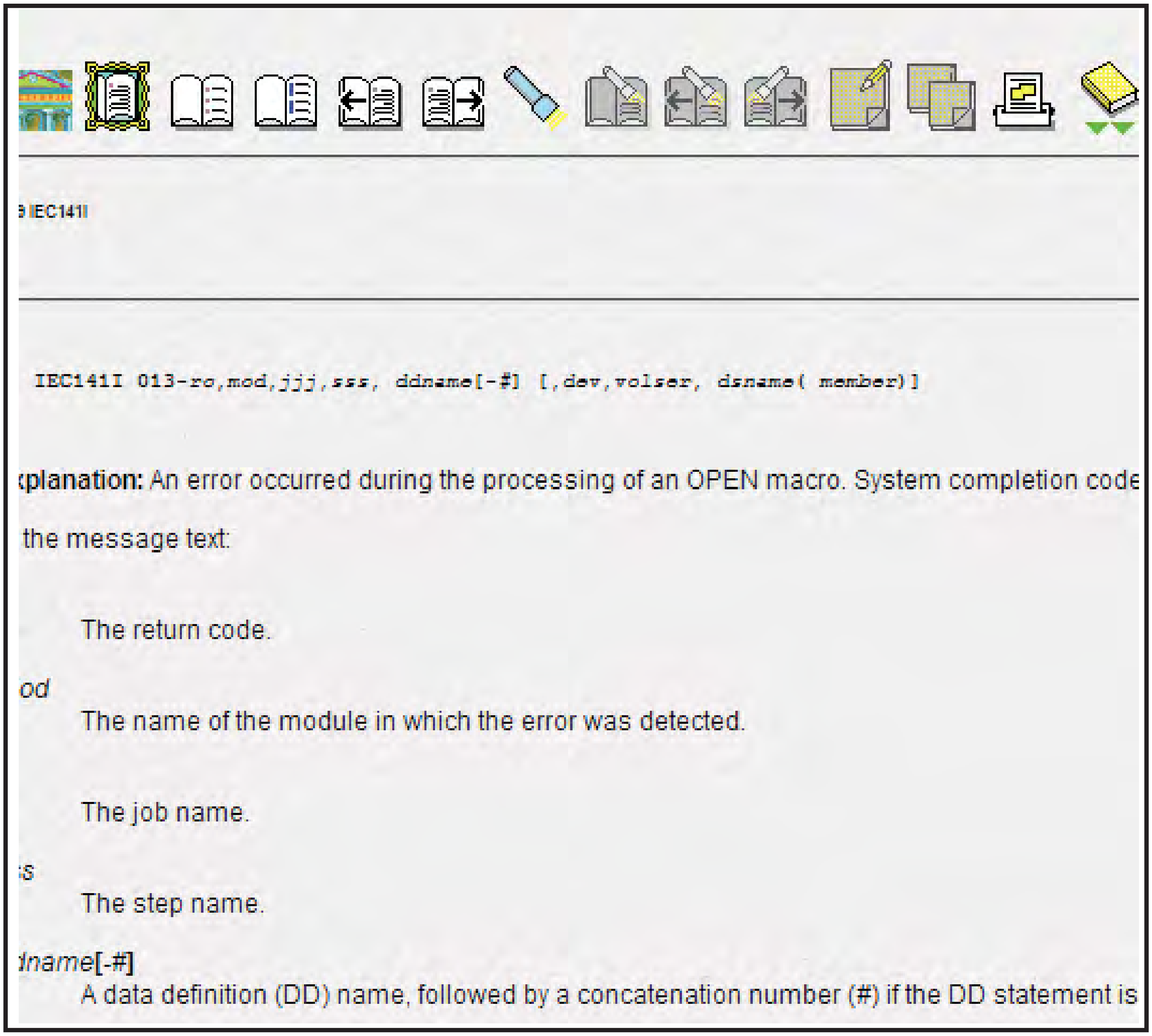
z/VM®
( ) V6R2
( ) V6R1
( ) V5R4

z/VSE™ and **VSE/ESA™**
( ) V5R1
( ) V4R1

- **Key in the message id, select an operating system and version, and click on "Go"...**

IBM Web-based Information, continued

❒ **As an experiment, we entered message id "iec141i", selected z/OS V1R13, and clicked Go, which took us to this page:**



3 IEC141I

```
IEC141I 013-rc,mod,jjj,sss, ddname[-#] [,dev,volser, dsname( member)]
```

xplanation: An error occurred during the processing of an OPEN macro. System completion code

the message text:

The return code.

od

The name of the module in which the error was detected.

The job name.

s

The step name.

dname[-#]

A data definition (DD) name, followed by a concatenation number (#) if the DD statement is

This page intentionally left almost blank.

BookManager

❒ **IBM Publications are available in machine readable format for many platforms (z/OS, AS/400, Windows, z/VSE, z/VM, AIX/6000)**

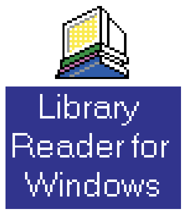
- **BookManager READ is a program product for accessing the publications at your terminal or workstation**

❒ **BookManager publications are organized under two major groupings**

- **Books - all publications, in sequence by publication number**

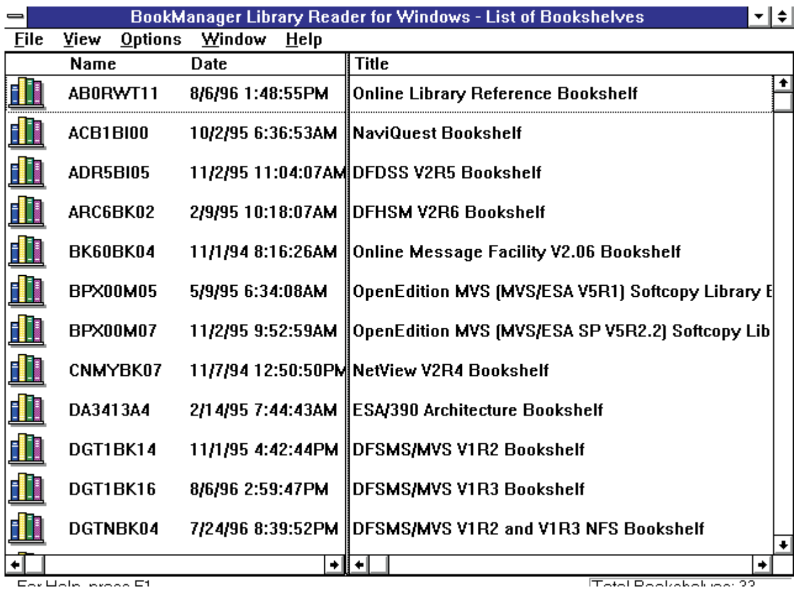
- **Bookshelf Lists - groupings by CD-ROM or product or related products**

❒ **Under Windows, you start BookManager READ by clicking on the correct icon:**



BookManager, continued

❒ The list of bookshelves looks like this



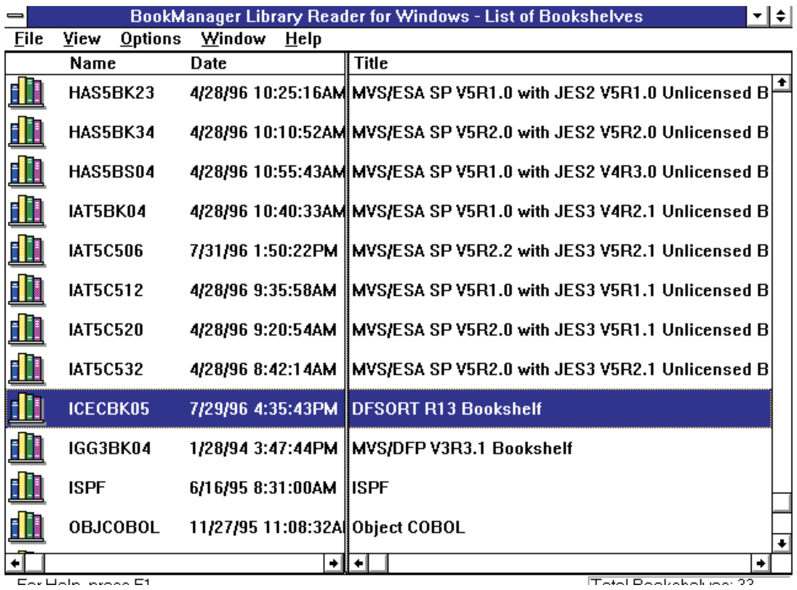
BookManager Library Reader for Windows - List of Bookshelves

File View Options Window Help

Name	Date	Title
AB0RWT11	8/6/96 1:48:55PM	Online Library Reference Bookshelf
ACB1BI00	10/2/95 6:36:53AM	NaviQuest Bookshelf
ADR5BI05	11/2/95 11:04:07AM	DFDSS V2R5 Bookshelf
ARC6BK02	2/9/95 10:18:07AM	DFHSM V2R6 Bookshelf
BK60BK04	11/1/94 8:16:26AM	Online Message Facility V2.06 Bookshelf
BPX00M05	5/9/95 6:34:08AM	OpenEdition MVS (MVS/ESA V5R1) Softcopy Library E
BPX00M07	11/2/95 9:52:59AM	OpenEdition MVS (MVS/ESA SP V5R2.2) Softcopy Lib
CNMYBK07	11/7/94 12:50:50PM	NetView V2R4 Bookshelf
DA3413A4	2/14/95 7:44:43AM	ESA/390 Architecture Bookshelf
DGT1BK14	11/1/95 4:42:44PM	DFSMS/MVS V1R2 Bookshelf
DGT1BK16	8/6/96 2:59:47PM	DFSMS/MVS V1R3 Bookshelf
DGTNBK04	7/24/96 8:39:52PM	DFSMS/MVS V1R2 and V1R3 NFS Bookshelf

For Help, press F1 Total Bookshelves: 33

❒ Double-click on a bookshelf to begin your explorations:



BookManager Library Reader for Windows - List of Bookshelves

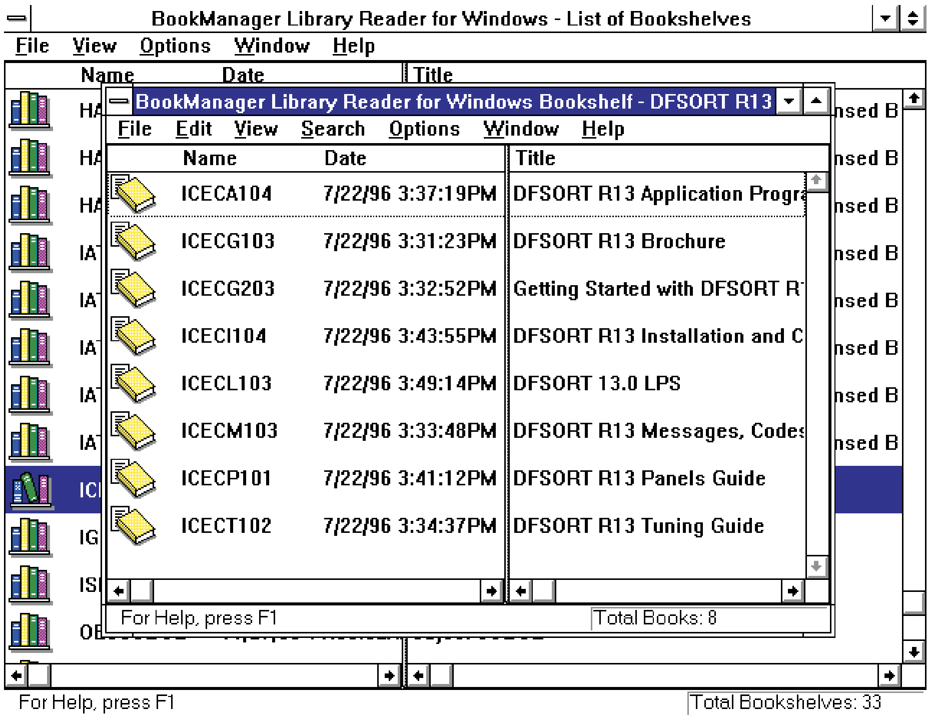
File View Options Window Help

Name	Date	Title
HAS5BK23	4/28/96 10:25:16AM	MVS/ESA SP V5R1.0 with JES2 V5R1.0 Unlicensed B
HAS5BK34	4/28/96 10:10:52AM	MVS/ESA SP V5R2.0 with JES2 V5R2.0 Unlicensed B
HAS5BS04	4/28/96 10:55:43AM	MVS/ESA SP V5R1.0 with JES2 V4R3.0 Unlicensed B
IAT5BK04	4/28/96 10:40:33AM	MVS/ESA SP V5R1.0 with JES3 V4R2.1 Unlicensed B
IAT5C506	7/31/96 1:50:22PM	MVS/ESA SP V5R2.2 with JES3 V5R2.1 Unlicensed B
IAT5C512	4/28/96 9:35:58AM	MVS/ESA SP V5R1.0 with JES3 V5R1.1 Unlicensed B
IAT5C520	4/28/96 9:20:54AM	MVS/ESA SP V5R2.0 with JES3 V5R1.1 Unlicensed B
IAT5C532	4/28/96 8:42:14AM	MVS/ESA SP V5R2.0 with JES3 V5R2.1 Unlicensed B
ICECBK05	7/29/96 4:35:43PM	DFSORT R13 Bookshelf
IGG3BK04	1/28/94 3:47:44PM	MVS/DFP V3R3.1 Bookshelf
ISPF	6/16/95 8:31:00AM	ISPF
OBJCOBOL	11/27/95 11:08:32AI	Object COBOL

For Help, press F1 Total Bookshelves: 33

BookManager, continued

❒ **You then get a list of books to choose from:**



BookManager Library Reader for Windows - List of Bookshelves

File View Options Window Help

BookManager Library Reader for Windows Bookshelf - DFSORT R13

File Edit View Search Options Window Help

Name	Date	Title
ICECA104	7/22/96 3:37:19PM	DFSORT R13 Application Progra
ICECG103	7/22/96 3:31:23PM	DFSORT R13 Brochure
ICECG203	7/22/96 3:32:52PM	Getting Started with DFSORT R
ICECI104	7/22/96 3:43:55PM	DFSORT R13 Installation and C
ICECL103	7/22/96 3:49:14PM	DFSORT 13.0 LPS
ICECM103	7/22/96 3:33:48PM	DFSORT R13 Messages, Codes
ICECP101	7/22/96 3:41:12PM	DFSORT R13 Panels Guide
ICECT102	7/22/96 3:34:37PM	DFSORT R13 Tuning Guide

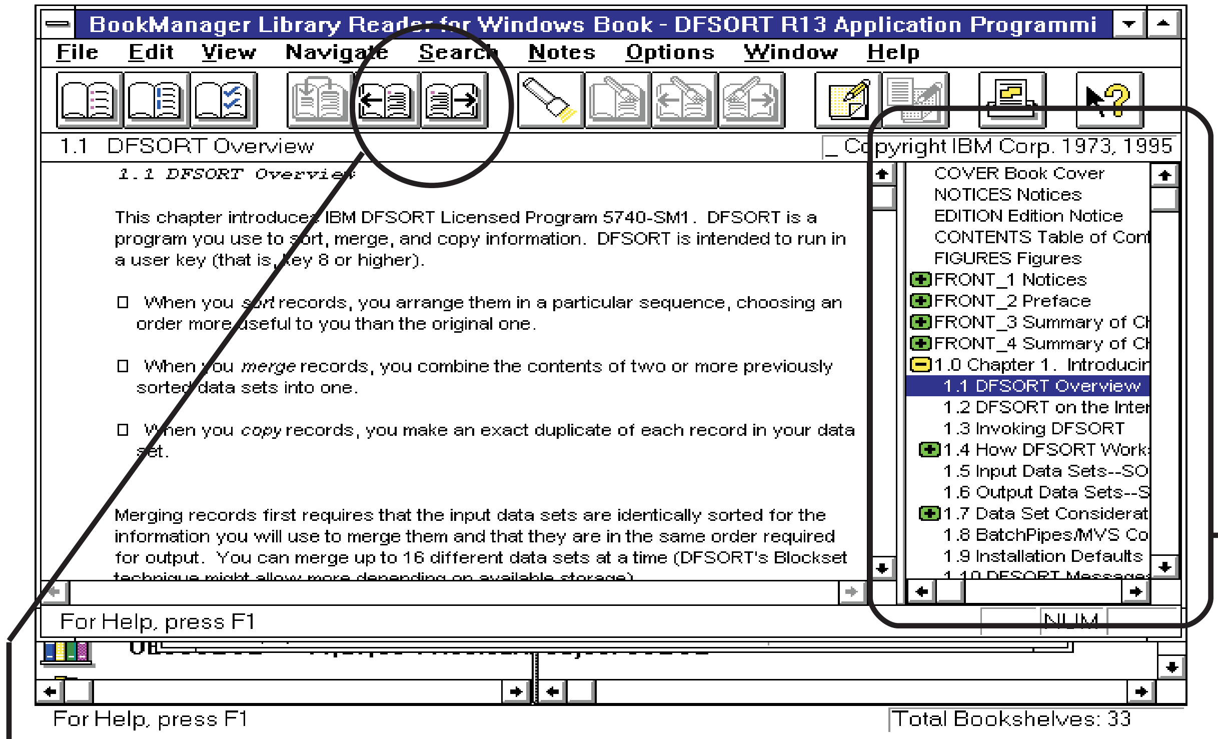
For Help, press F1 Total Books: 8

For Help, press F1 Total Bookshelves: 33

❒ **Then double-click on the book you want to reference**

- ♦ **This leads you to the information you're after**

❒ Main controls for BookManager READ:


BookManager Library Reader for Windows Book - DFSORT R13 Application Programmi
File Edit View Navigate Search Notes Options Window Help
1.1 DFSORT Overview
Copyright IBM Corp. 1973, 1995
For Help, press F1
NUM
For Help, press F1
Total Bookshelves: 33

1.1 DFSORT Overview

This chapter introduces IBM DFSORT Licensed Program 5740-SM1. DFSORT is a program you use to sort, merge, and copy information. DFSORT is intended to run in a user key (that is, key 8 or higher).

- When you *sort* records, you arrange them in a particular sequence, choosing an order more useful to you than the original one.
- When you *merge* records, you combine the contents of two or more previously sorted data sets into one.
- When you *copy* records, you make an exact duplicate of each record in your data set.

Merging records first requires that the input data sets are identically sorted for the information you will use to merge them and that they are in the same order required for output. You can merge up to 16 different data sets at a time (DFSORT's Blockset technique might allow more depending on available storage).

COVER Book Cover
NOTICES Notices
EDITION Edition Notice
CONTENTS Table of Cont
FIGURES Figures
FRONT_1 Notices
FRONT_2 Preface
FRONT_3 Summary of Ch
FRONT_4 Summary of Ch
1.0 Chapter 1. Introducin
1.1 DFSORT Overview
1.2 DFSORT on the Inter
1.3 Invoking DFSORT
1.4 How DFSORT Work
1.5 Input Data Sets--SO
1.6 Output Data Sets--S
1.7 Data Set Considerat
1.8 BatchPipes/MVS Co
1.9 Installation Defaults
1.10 DFSORT Messages

- ♦ **<u>File menu</u> - lets you print a topic, chapter, or the whole book; also lets you open a new book, or exit**
- ♦ **<u>Page Turner Controls</u> - step forward or backward a page at a time**
- ♦ **<u>Notes menu</u> - lets you take notes (and later retrieve them, change them, delete them)**
- ♦ **You can quickly move to chapters and sub-sections by using the scrollable box: clicks and double-clicks let you explore and navigate quickly**

Quick Reference

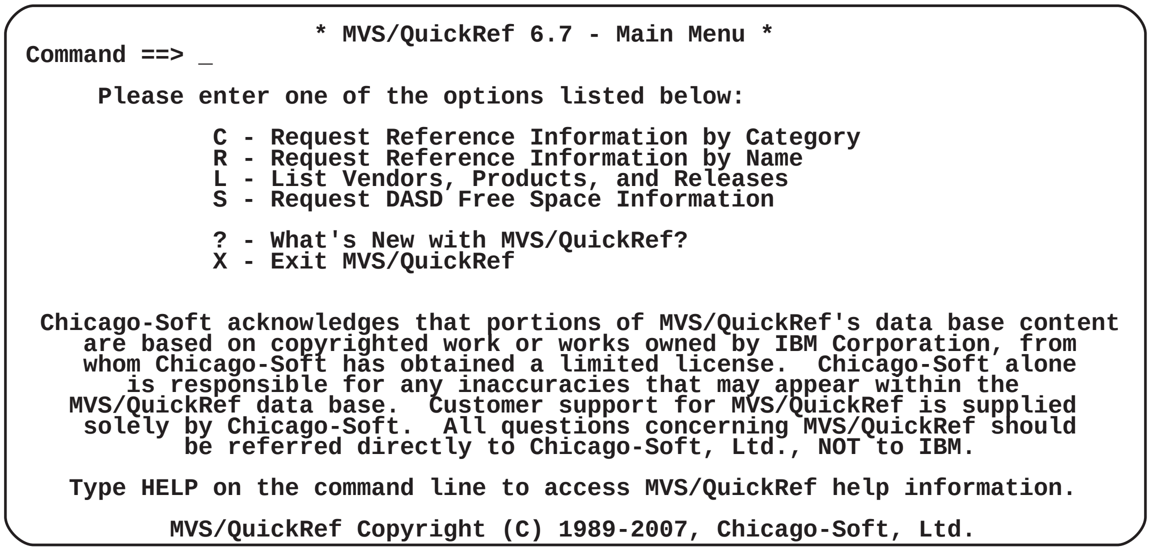
❒ **Quick Reference is a handy tool that lets you look up messages and codes from a variety of sources; many shops have this**

- ♦ **A quick way to find a description, along with suggested solutions if the message is an error message**

❒ **You get to Quick Reference by entering the string QW on any command / option line:**

```
Command ===> qw
```

- ♦ **First you'll see a panel indicating there is a lot of new stuff in the current release; when you press <Enter> you'll see the the Main Menu:**



```
                    * MVS/QuickRef 6.7 - Main Menu *
Command ==> _

     Please enter one of the options listed below:

             C - Request Reference Information by Category
             R - Request Reference Information by Name
             L - List Vendors, Products, and Releases
             S - Request DASD Free Space Information

             ? - What's New with MVS/QuickRef?
             X - Exit MVS/QuickRef


 Chicago-Soft acknowledges that portions of MVS/QuickRef's data base content
    are based on copyrighted work or works owned by IBM Corporation, from
    whom Chicago-Soft has obtained a limited license.  Chicago-Soft alone
       is responsible for any inaccuracies that may appear within the
   MVS/QuickRef data base.  Customer support for MVS/QuickRef is supplied
    solely by Chicago-Soft.  All questions concerning MVS/QuickRef should
           be referred directly to Chicago-Soft, Ltd., NOT to IBM.

   Type HELP on the command line to access MVS/QuickRef help information.

          MVS/QuickRef Copyright (C) 1989-2007, Chicago-Soft, Ltd.
```

- ♦ **If you choose option R, you are prompted for a vendor name and, optionally, a product name and release number**
  - ✗ This leads to a list of products for the selected vendor for which there is information available under Quick Reference

Quick Reference, 2

❐ **When you select option R, you see this panel:**

```
            * MVS/QuickRef 6.7 - Request Reference Information *
Command ==> _

Enter keys for the desired item.  Each key may be entered as a full key or as
a generic key prefix ending in '*'.  A key which is entered as a single '*'
(or which is left blank) is treated as a match.

  Vendor ==>                         (Vendor to be searched)
  Product ==>                        (Product to be searched)
  Release ==>                        (Release of product to be searched)
  Item ==>                           (Message ID, abend code, command, etc.)

For example, type just an item name in the Item field to search all vendors,
products, and releases for that item.  Type IBM for Vendor and IEF* for Item
to search all releases of all IBM products for item names starting with 'IEF'.
Type '?' for Vendor to transfer to the List Vendors/Products/Releases panel.



  Type HELP on the command line to access MVS/QuickRef help information.

         MVS/QuickRef Copyright (C) 1989-2007, Chicago-Soft, Ltd.
```

❐ **There are a variety of ways to proceed from this panel, but here we focus on getting information about error messages**

- ♦ **Place IBM in the Vendor input field: Vendor ==> ibm**
  - ✗ Then a message id in the Item input field
    - ➢ In its entirety: Item ==> IEC141I
    - ➢ Or with a wild card: Item ==> IEC14*
  - ✗ Or, place a system completion code in the Item input field (as a letter S followed by the code)
    - ➢ In its entirety: Item ==> S014
    - ➢ Or with a wild card: Item ==> S2*

Quick Reference, 3

❒ **If you entered a message id or completion code completely, you are placed in browse of the corresponding description**

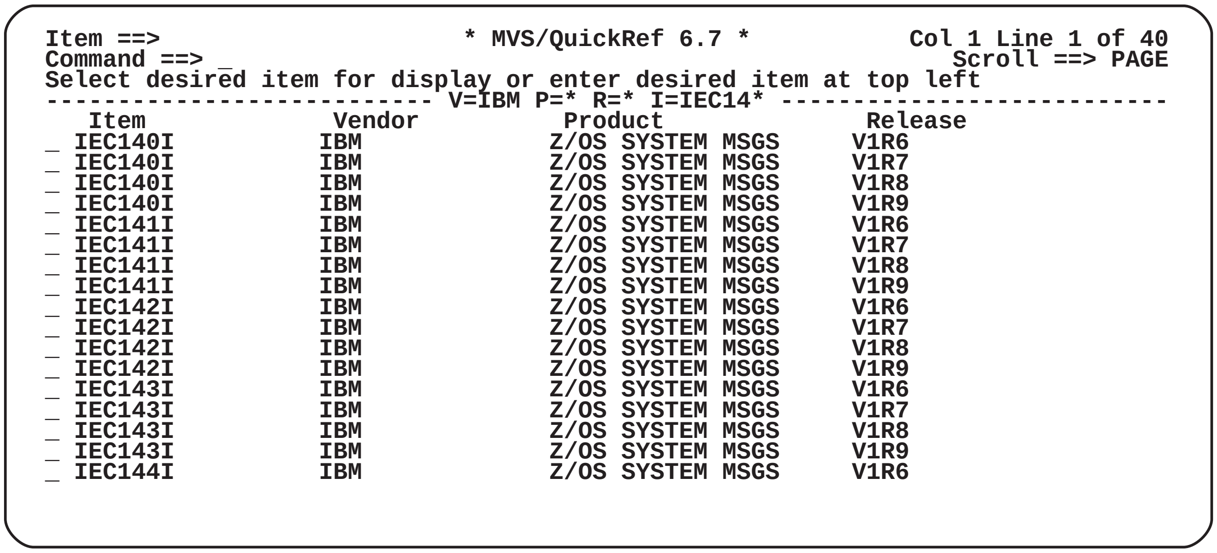
- ♦ **We look at this in just a moment**

❒ **If your request included a wild card (*), you will first see a screen showing all the choices that satisfied your search request**

- ♦ **For example, if you entered**

```
Vendor ==> ibm
Product ===>
Release ===>
Item ==> iec14*
```

- ♦ **You would see this screen:**

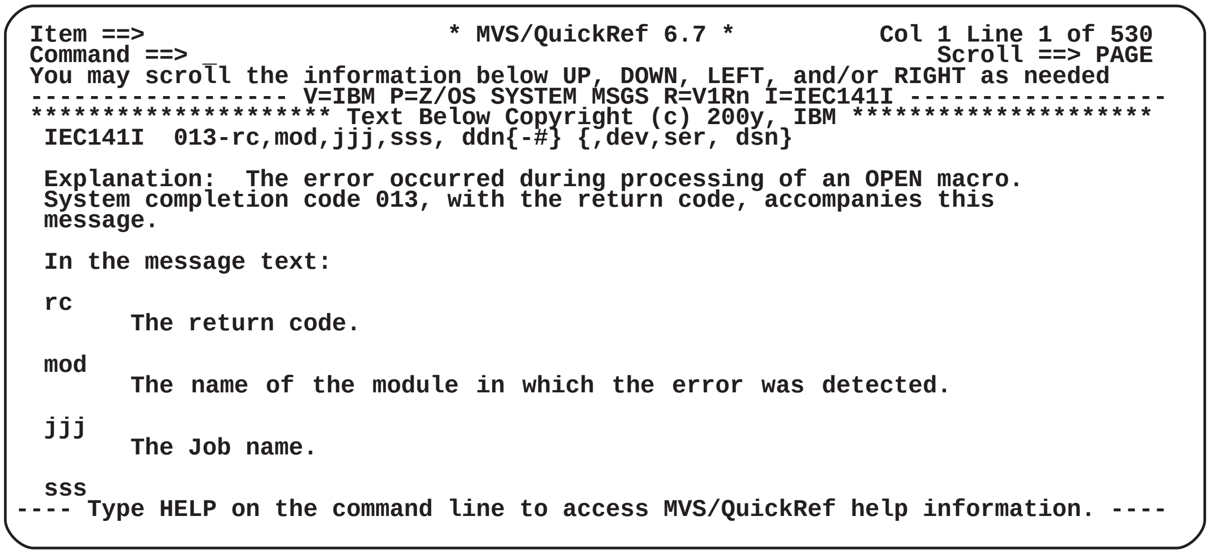


```
Item ==>                    * MVS/QuickRef 6.7 *          Col 1 Line 1 of 40
Command ==> _                                                 Scroll ==> PAGE
Select desired item for display or enter desired item at top left
---------------------------- V=IBM P=* R=* I=IEC14* ---------------------------
   Item             Vendor           Product              Release
_ IEC140I          IBM             Z/OS SYSTEM MSGS     V1R6
_ IEC140I          IBM             Z/OS SYSTEM MSGS     V1R7
_ IEC140I          IBM             Z/OS SYSTEM MSGS     V1R8
_ IEC140I          IBM             Z/OS SYSTEM MSGS     V1R9
_ IEC141I          IBM             Z/OS SYSTEM MSGS     V1R6
_ IEC141I          IBM             Z/OS SYSTEM MSGS     V1R7
_ IEC141I          IBM             Z/OS SYSTEM MSGS     V1R8
_ IEC141I          IBM             Z/OS SYSTEM MSGS     V1R9
_ IEC142I          IBM             Z/OS SYSTEM MSGS     V1R6
_ IEC142I          IBM             Z/OS SYSTEM MSGS     V1R7
_ IEC142I          IBM             Z/OS SYSTEM MSGS     V1R8
_ IEC142I          IBM             Z/OS SYSTEM MSGS     V1R9
_ IEC143I          IBM             Z/OS SYSTEM MSGS     V1R6
_ IEC143I          IBM             Z/OS SYSTEM MSGS     V1R7
_ IEC143I          IBM             Z/OS SYSTEM MSGS     V1R8
_ IEC143I          IBM             Z/OS SYSTEM MSGS     V1R9
_ IEC144I          IBM             Z/OS SYSTEM MSGS     V1R6
```

❒ **Select the message you are interested in by placing the cursor next to the message for the release you are working with and pressing <Enter>**

Quick Reference, 4

❒ **When you get to a message description, it looks like this:**



```
Item ==>                     * MVS/QuickRef 6.7 *          Col 1 Line 1 of 530
Command ==> _                                                  Scroll ==> PAGE
You may scroll the information below UP, DOWN, LEFT, and/or RIGHT as needed
------------------ V=IBM P=Z/OS SYSTEM MSGS R=V1Rn I=IEC141I ------------------
********************* Text Below Copyright (c) 200y, IBM *********************
 IEC141I  013-rc,mod,jjj,sss, ddn{-#} {,dev,ser, dsn}

 Explanation:  The error occurred during processing of an OPEN macro.
 System completion code 013, with the return code, accompanies this
 message.

 In the message text:

 rc
       The return code.

 mod
       The name of the module in which the error was detected.

 jjj
       The Job name.

 sss
---- Type HELP on the command line to access MVS/QuickRef help information. ----
```

♦ **Notice the message format**

✗ In Quick Reference notation, **ddn** means **DDname**, **ser** means **volume serial**, **dsn** means **data set name**

♦ **Scroll up and down as usual under ISPF to read the entire information about the topic at hand**

♦ **If there some suggested action, it is always found at the bottom of a description**

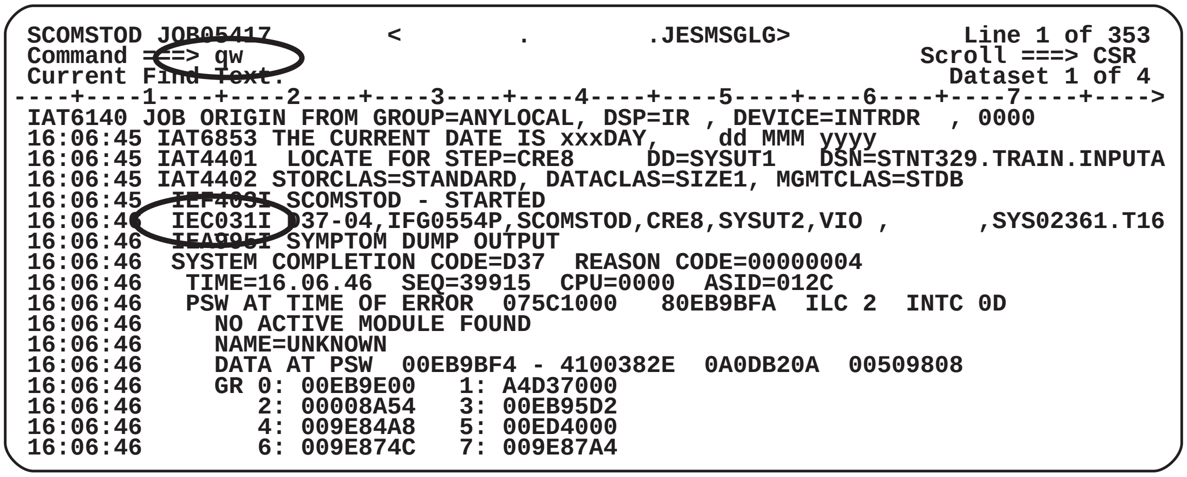
Quick Reference and Job Listings

❐ **There is a nice relationship between Quick Reference and the job listings from (E)JES, SDSF, IOF, Flasher, and so on that can speed up your message / error code look up work**

❐ **When you are looking at your JCL listing under (E)JES, say, if there is a message or completion code on the screen you want to look up, code "qw" on the command line:**

```
Command ===> qw
```

- ♦ **Then, before you press <Enter>, move the cursor to any position on the message id or system completion code:**

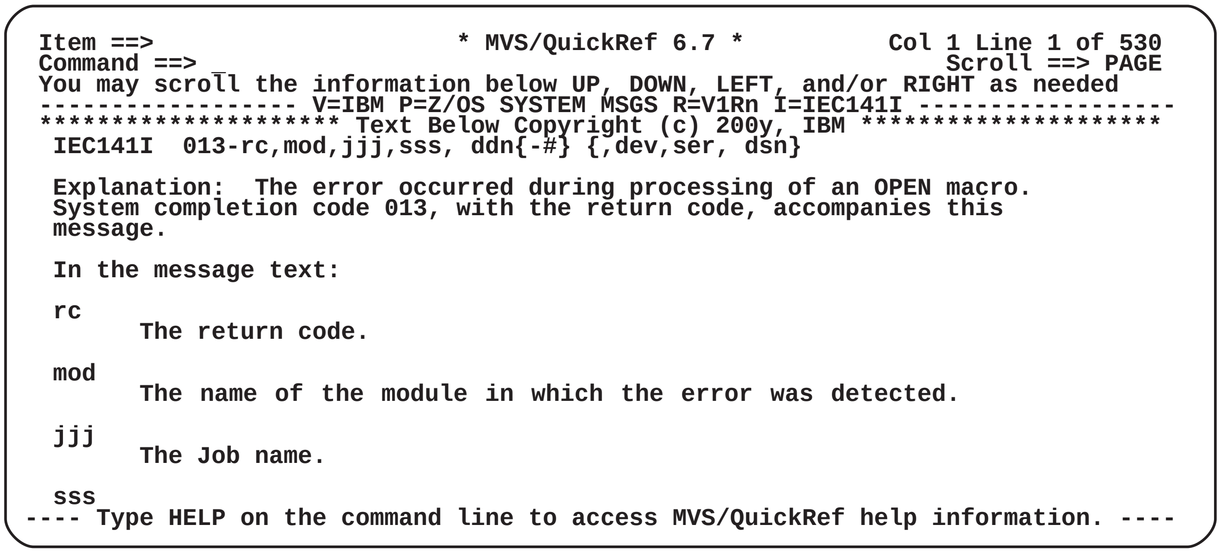


```
 SCOMSTOD JOB05417        <        .        .JESMSGLG>          Line 1 of 353
 Command ===> qw                                              Scroll ===> CSR
 Current Find Text:                                              Dataset 1 of 4
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+---->
 IAT6140 JOB ORIGIN FROM GROUP=ANYLOCAL, DSP=IR , DEVICE=INTRDR  , 0000
 16:06:45 IAT6853 THE CURRENT DATE IS xxxDAY,    dd MMM yyyy
 16:06:45 IAT4401  LOCATE FOR STEP=CRE8     DD=SYSUT1   DSN=STNT329.TRAIN.INPUTA
 16:06:45 IAT4402 STORCLAS=STANDARD, DATACLAS=SIZE1, MGMTCLAS=STDB
 16:06:45  IEF403I SCOMSTOD - STARTED
 16:06:46  IEC031I D37-04,IFG0554P,SCOMSTOD,CRE8,SYSUT2,VIO ,      ,SYS02361.T16
 16:06:46  IEA995I SYMPTOM DUMP OUTPUT
 16:06:46  SYSTEM COMPLETION CODE=D37  REASON CODE=00000004
 16:06:46   TIME=16.06.46  SEQ=39915  CPU=0000  ASID=012C
 16:06:46   PSW AT TIME OF ERROR  075C1000   80EB9BFA  ILC 2  INTC 0D
 16:06:46     NO ACTIVE MODULE FOUND
 16:06:46     NAME=UNKNOWN
 16:06:46     DATA AT PSW  00EB9BF4 - 4100382E  0A0DB20A  00509808
 16:06:46     GR 0: 00EB9E00   1: A4D37000
 16:06:46        2: 00008A54   3: 00EB95D2
 16:06:46        4: 009E84A8   5: 00ED4000
 16:06:46        6: 009E874C   7: 009E87A4
```

- ♦ **Then press Enter, and you are jumped to the error description page appropriate to the error**

Quick Reference and Job Listings, 2

- The appropriate error message should help in solving problems



```
Item ==>                     * MVS/QuickRef 6.7 *         Col 1 Line 1 of 530
Command ==> _                                                  Scroll ==> PAGE
You may scroll the information below UP, DOWN, LEFT, and/or RIGHT as needed
------------------ V=IBM P=Z/OS SYSTEM MSGS R=V1Rn I=IEC141I -----------------
******************** Text Below Copyright (c) 200y, IBM ********************
 IEC141I  013-rc,mod,jjj,sss, ddn{-#} {,dev,ser, dsn}

 Explanation:  The error occurred during processing of an OPEN macro.
 System completion code 013, with the return code, accompanies this
 message.

 In the message text:

 rc
       The return code.

 mod
       The name of the module in which the error was detected.

 jjj
       The Job name.

 sss
---- Type HELP on the command line to access MVS/QuickRef help information. ----
```

- This technique can be very helpful in solving JCL and program abend problems

Section Preview

❒ **COND Parameter**

- **Although this parameter has been obsolete for over 15 years, it is still found in JCL today - this section covers the syntax and use for reference and maintenance purposes**

Condition Code Testing

❒ **Condition codes returned from any given step in a job may be tested in any subsequent step, using the COND parameter of the EXEC statement:**

```
//STEP12    EXEC    PGM=xxxxxxxx,COND=(n,op,stepname)
```

n	—	**value to compare to**
op	—	**comparison operator; one of**

EQ	**(for equal)**
NE	**(for not equal)**
GT	**(for greater than)**
LT	**(for less than)**
LE	**(for less than or equal)**
GE	**(for greater than or equal)**

<u>**IF THE CONDITION IS TRUE, DO NOT RUN THIS STEP**</u>

<u>**For example:**</u>

COND=(5,LE,STEPXX)

Condition code from <u>Step named STEPXX</u>	<u>Run current step? (Yes or No)</u>
0	___
4	___
5	___
6	___

Multiple Condition Tests

❒ **Up to 8 COND tests can be made on a single EXEC statement**

```
//STEPXT      EXEC     PGM=ISDRUSPT,
//          COND=((4,LE,INSPCT),(8,NE,TRSN),
//          (5,GT,TRNSPL),(8,LT,TRNSPL))
```

❒ **If <u>any one</u> (or more than one) of these conditions is true, do not run this step**

Special COND Tests

<u>**COND=EVEN**</u>

- ♦ **Run this step <u>even if some previous step has ABENDed!</u>**

<u>**COND=ONLY**</u>

- ♦ **Run this step <u>only if some previous step has ABENDed!</u>**

❒ **ONLY and EVEN are mutually exclusive**

- ♦ **Either one may be combined with up to 7 of the other kind of COND tests:**

```
COND=((5,GE,TESTR),(5,GE,RESTR),EVEN)
```

❒ **COND is evaluated for EVEN and ONLY first**

- ♦ **If the step should run based on this, then any other COND tests are evaluated to determine if the step should run**

COND With No Stepname

❒ **You may specify COND in the form:**

```
COND=(5,LE)
```

- ♦ **Then if <u>any</u> preceding step meets the COND test, this step will not run**

❒ **You may also have multiple tests of this form, and mix this form of test with the other form and with EVEN or ONLY**

COND on the JOB Statement

❒ **A COND test may be specified on the JOB statement, if it is of the form with no stepname:**

```
//QRSKKL        JOB        ...,COND=(5,LT)
```

or

```
//KRREF         JOB        ...,COND=((50,LT),(3,EQ))
```

❒ **Thus if <u>any</u> step causes <u>any</u> COND test to be true,**

<u>the rest of the job is bypassed</u>

General Caution Regarding COND

- **If you skip a step due to COND testing, be aware that later steps may not find data sets the skipped step was supposed to create**

  - **Or duplicate data set names can occur if this step was supposed to delete some data sets**

This page intentionally left almost blank.

Summary of ISPF Commands Discussed In "ISPF and JCL on z/OS"

General ISPF Commands

ACTIONS
CANCEL
COLOR
CMDE
CUAATTR
END
EPDF *dsname***[(***membername***)] [BROWSE | VIEW]**
EXIT
FKA [ON|SHORT|OFF|PREFIX|NOPREFIX]
HELP
KEYLIST {ON|OFF}
KEYS
KEYSHELP
LIST [PRINT|DELETE|KEEP]
LOG [PRINT|DELETE|KEEP]
PFSHOW [ON|OFF|TAILOR]
PRINT
PSCOLOR
REFACTD *list_name* **[***n***]**
REFACTL *list_name* **[***n***]**
REFADDD *list_name*
REFADDL *list_name*
REFLISTD [*n***]**
REFLISTL [*n***]**
REFOPEND
REFOPENL
RETF
RETP
RETRIEVE
RETURN
SCRNAME {*name* **[PERM] | [ON|OFF]}**
SETTINGS
SPLIT [NEW]
SWAP [LIST|PREV|NEXT|*scrname***|***scrnid***]**
SWAPBAR {ON|OFF}
SYSNAME {ON | OFF}
TSO *command*
USERID {ON | OFF}
WS *command*
ZKEYS

ISPF Positioning Commands

BOTTOM (synonym)
DOWN
LEFT
Locate (in member lists)
RIGHT
Select (in some member lists)
UP
TOP (synonym)

ISPF Edit / View Primary non-profile Commands

BOUnds [*left_col***] [***right_col***]**
BROWSE [*membername***]**
CANcel
Change {*string_1***|*} {***string_2***|*} [***label_1 label_2***]**
[X|NX] [NEXT|ALL|FIRST|LAST|PREV]
[CHARs|PREfix|SUFfix|WORD]
[*col_1***] [***col_2***]**
COL[S]
COPY [[*dsn***][(***membername***)]][{AFTer|BEFore}** *label***]**
CREate [[(*membername***)]] [***label_1 label_2***]**
CUT [*line_range***] [***clipboard***] [REPLACE|APPEND] [DISPLAY]**
DELete ALL [X|NX] [*label_1 label_2***]**
EDIT [*membername***]**
EDITSET
eXclude *string* **[NEXT|ALL|FIRST|LAST|PREV]**
[CHARs|PREfix|SUFfix|WORD]
[*col_1***] [***col_2***] [***label_1 label_2***]**
Find {*string***|*} [X|NX] [NEXT|ALL|FIRST|LAST|PREV]**
[CHARs|PREfix|SUFfix|WORD]
[*col_1***] [***col_2***] [***label_1 label_2***]**
FLIP
HIDE X
LEVel *n*
Locate {*line_no* **|** *label***}**
Locate [FIRST|LAST|NEXT|PREV]
{CHANGE|COMmand|ERRor|
EXCLUDED|SPECIAL}
[*label_1 label_2***]**
MOVE [*dsn***[(***membername***)]] [{AFTer|BEFore}** *label***]**
NONUMber
PASTE [*clipboard***] [{AFTER | BEFORE}** *label***][KEEP|DELETE]**
PRESERVE {ON|OFF}
RCHANGE
RENumber {OFF | ON [STD] [COBOL]}
REPlace [*dsn***[(***membername***)]] [***label_1 label_2***]**
RESet [LABel]
RFIND
SAVE
SORT [*label_1 label_2***] [X|NX] [{A|D} [***col_1***] [***col_2***]...]**
SUBmit
UNDO
UNNumber
VERsion *n*
VIEW [*membername***]**

Tutorial Commands

BACK
INDEX
SKIP
TOC
UP

Summary of ISPF Commands Discussed In "ISPF and JCL on z/OS", p. 2

ISPF Edit / View Primary Profile Commands

AUTOSAVE	**{ON|OFF [PROMPT|NOPROMPT]}**
AUTONUM	**{ON|OFF}**
AUTOLIST	**{ON|OFF}**
CAPS	**{ON|OFF}**
HIlite	**[ON|OFF|[{DO|IF|NO}LOGIC]**
	[AUTO|DEFAULT|OTHER|ASM|BOOK|
	C|COBOL|DTL|IDL|JCL|PANEL|PLI|
	REXX|SKEL]
	[RESET] [PAREN] [FIND] [CURsor]
	[SEARCH] [DISABLED]
HEX	**{ON [VERT|DATA] | OFF}**
NULLs	**{ON [STD|ALL] | OFF}**
NUMber	**{ON {COBOL|STD} | OFF}**
PACK	**{ON|OFF}**
PRofile	**{***name***|LOCK|UNLOCK|RESET|***n***}**
RECovery	**{ON|OFF [WARN|NOWARN]}**
SETUNDO	**{STORAGE | RECOVERY | OFF|KEEP}**
STATS	**{ON|OFF}**
TABs	**{ON [***tab_char***] [STD|ALL] | OFF}**

ISPF Edit / View Line Commands

) - right column shift	**)**	**)***n*	**))...))**	**))...))***n*
(- left column shift	**(**	**(***n*	**((...((**	**((...((***n*
> - right data shift	**>**	**>***n*	**>>...>>**	**>>...>>***n*
< - left data shift	**<**	**<***n*	**<<...<<**	**<<...<<***n*
A - after (for Move or Copy)			**A**	**A***n*
AK - after and keep (for M & C)			**AK**	**AK***n*
B - before (for Move or Copy)			**B**	**B***n*
BK - before and keep (for M & C)			**BK**	**BK***n*
BNDs - display a bounds line				
C - copy:	**C**	**C***n*	**CC...CC**	
COLs - insert columns line				
D - delete	**D**	**D***n*	**DD...DD**	
F - first (for excluded lines)			**F**	**F***n*
HX - display line in Hex		**HX**	**HX***n*	
L - last (for excluded lines)			**L**	**L***n*
LC - lower case	**LC**	**LC***n*	**LCC...LCC**	
M - move	**M**	**M***n*	**MM...MM**	
MASK - display a mask line				
MD - make line a data line		**MD**	**MD***n*	
		MDMD.. MDMD		
O - overlay (for Move or Copy)			**O** **O***n*	**OO...OO**
OK - overlay and keep	**OK**	**OK***n*		**OOK - - - OOK**
R - repeat	**R**	**R***n*	**RR...RR**	**RR...RR***n*
S - show (for excluded lines)			**S** **S***n*	
TABs - display a tabs line				
TS - text split	**TS**			
UC - upper case	**UC**	**UC***n*	**UCC...UCC**	
X - exclude lines	**X**	**X***n*	**XX...XX**	

Memberlist Line Commands
(depending on which member list)

/ - select (1, 2, 3.1, 3.4)
B - browse (1, 3.1, 3.3, 3.4, 11)
C - copy (3.1, 3.4, 11)
D - delete (3.1, 3.4, 11)
E - edit (1, 2, 3.1, 3.4, 11)
G - reset statistics (3.1, 3.4, 11)
J - submit (3.1, 3.4, 11)
M - move (3.1, 3.4)
MO - move (11)
P - print (3.1, 3.4, 11)
R - rename (3.1, 3.4, 11)
S - select (1, 2, 3.3, 11)
SUB - submit (3.4)
T - tso command (3.1, 3.4, 11)
V - view (1, 2, 3.1, 3.4, 11)
W - workstation command (3.1, 3.4, 11)

Memberlist Primary Commands

MLC
REFRESH
SORT [*col_name* **[{ A | D }]] [***col_name* **[{ A | D }]]**

DSLIST Line Commands (subset)

B - browse
C - catalog
CO - copy
D - delete
E - edit
F - free unused space
I - data set information
M - memberlist
MO - move
P - print
PX - print index (directory)
R - rename
V - view
Z - compress data set

DSLIST Primary Commands (subset)

APPEND [*list_name***|***dsname_level***]**
MEMBER *string* **[X|NX] [RECALL1|RECALL2]**
SAVE [*name***]**
SORT *column_name*
VA, VS, VT, VV - change view

Summary Of JCL Statements Examined In This Course

//jobname	JOB	(accounting_info),programmer_name	
		CLASS	
		JOBRC	
		MEMLIMIT	
		MSGCLASS	
		NOTIFY	
		REGION	
		SCHENV	
		TIME	(min,sec)
		TYPRUN	{SCAN|HOLD}
//stepname	EXEC	{PGM=|PROC=|procname}	
		MEMLIMIT	
		PARM	
		REGION	
		TIME	(min,sec)
//ddname	DD	{*|DATA}	
		DLM	
//ddname	DD	SYSOUT	(class[,writer][,forms])
		BLKSIZE	
		COPIES	
		DEST	
		FREE	
		HOLD	
		LRECL	
		OUTLIM	
		OUTPUT	
		RECFM	
		SPIN	
		SEGMENT	

Summary Of JCL Statements Examined In This Course, 2

```
//ddname     DD         DUMMY
                        AVGREC
                        BLKSIZE
                        DATACLAS
                        DISP                 - see next page
                        {DSN | DSNAME}
                        DSNTYPE
                        EXPDT
                        KEYOFF
                        LABEL                (file_no,label_type)
                        LIKE
                        LRECL
                        MGMTCLAS
                        RECFM
                        RECORG
                        RETPD
                        SPACE
                            (unit,(pri[,sec][,dir])[,RLSE])
                        STORCLAS
                        TRTCH
                        UNIT                {(dev[,count])
                        VOL                 {SER
                                             (,RETAIN)
```

Summary Of JCL Statements Examined In This Course, 3

DISP parameter subparameters:

NEW	**,KEEP**	**,KEEP**
MOD	**,DELETE**	**,DELETE**
OLD	**,CATLG**	**,CATLG**
SHR	**,UNCATLG**	**,UNCATLG**
	,PASS	

//outpname	**OUTPUT**	**CLASS**
		COPIES
		DEFAULT
		DEST
		JESDS
		PRTY
		NOTIFY
		OUTDISP

//procname PROC [default values for symbolics]

//* — Comment Statement

// — Null Statement

/* — SYSIN data delimiter

Summary Of JCL Statements Examined In This Course, 4

```
//ifname      IF      relational_condition   THEN

//elsename    ELSE

//endifname   ENDIF

//setname     SET         symbolic=value[,...]

//inname      INCLUDE     MEMBER=membername

//jcllibname  JCLLIB      ORDER=(dsname[,...])
```

Index

This page intentionally left almost blank.

JCL Skeletons

❒ To speed the development of coding JCL, it is sometimes helpful to have a set of pre-coded JCL members in your library that you can simply copy into your new JCL using the editor

❒ For example, suppose you had the following pre-coded members in your library:

- JOB - a JOB statement all set with your most common set of parameters, say ...

```
//E0145   JOB  (123,000-99),COMSTOCK,CLASS=A,
//    NOTIFY=E0145,MSGCLASS=X
```

- JOBLIB - a DD statement with a DDname of JOBLIB identifying a load library; might have several DD statements concatenated under the single DD name, for example ...

```
//JOBLIB   DD  DSN=OUR.PRODCTN.LOADLIB,DISP=SHR
//         DD  DSN=OUR.TEST.LOADLIB,DISP=SHR
```

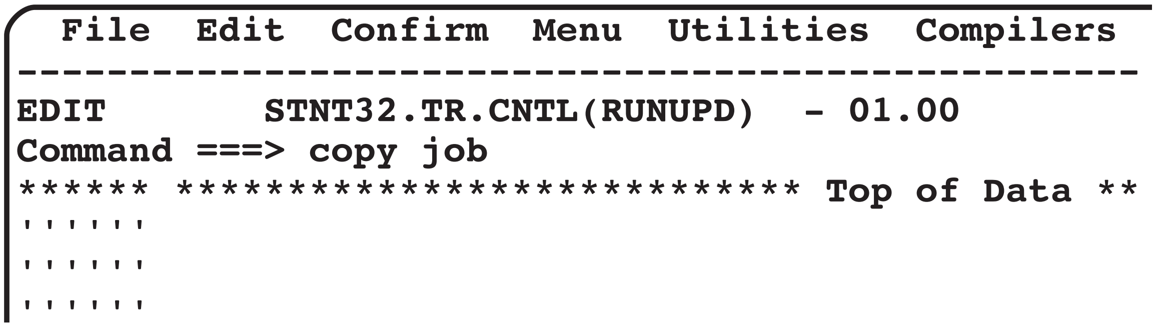
- EXEC - an EXEC statement with skeleton parameters

```
//STEPNN   EXEC  PGM=------     ,PARM=
```

- And others, of course

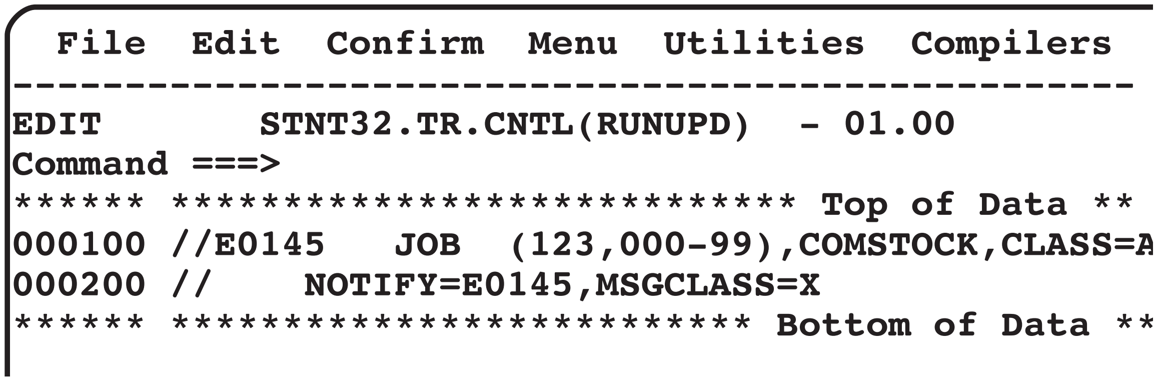
❒ **Now, when you start coding, you have an empty screen; copy in the JOB member:**

Before



```
  File  Edit  Confirm  Menu  Utilities  Compilers
--------------------------------------------------
EDIT       STNT32.TR.CNTL(RUNUPD)  - 01.00
Command ===> copy job
****** **************************** Top of Data **
''''''
''''''
''''''
```

After



```
  File  Edit  Confirm  Menu  Utilities  Compilers
--------------------------------------------------
EDIT       STNT32.TR.CNTL(RUNUPD)  - 01.00
Command ===>
****** **************************** Top of Data **
000100 //E0145   JOB  (123,000-99),COMSTOCK,CLASS=A
000200 //    NOTIFY=E0145,MSGCLASS=X
****** ************************** Bottom of Data **
```

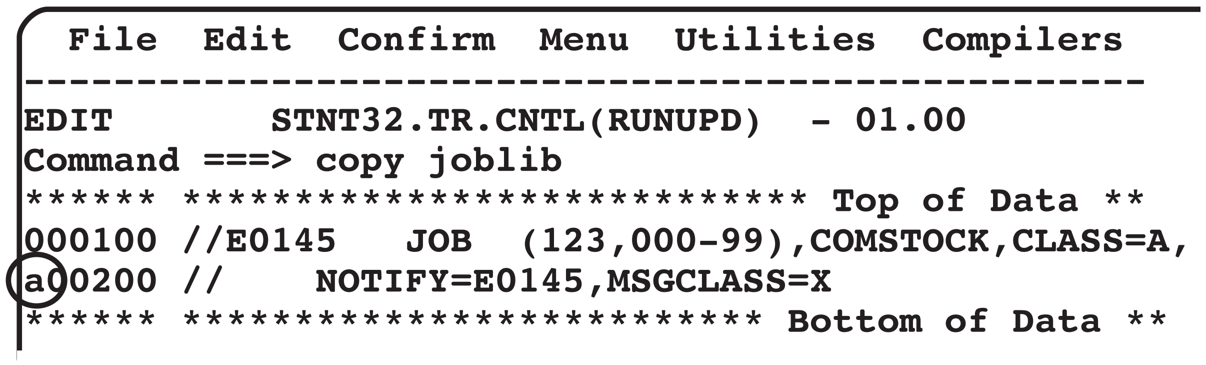
JCL Skeletons, continued

❐ Now, to copy in the next member, you will have to tell the editor where to place the new text

- Place an 'a' (for 'after') in the sequence number column

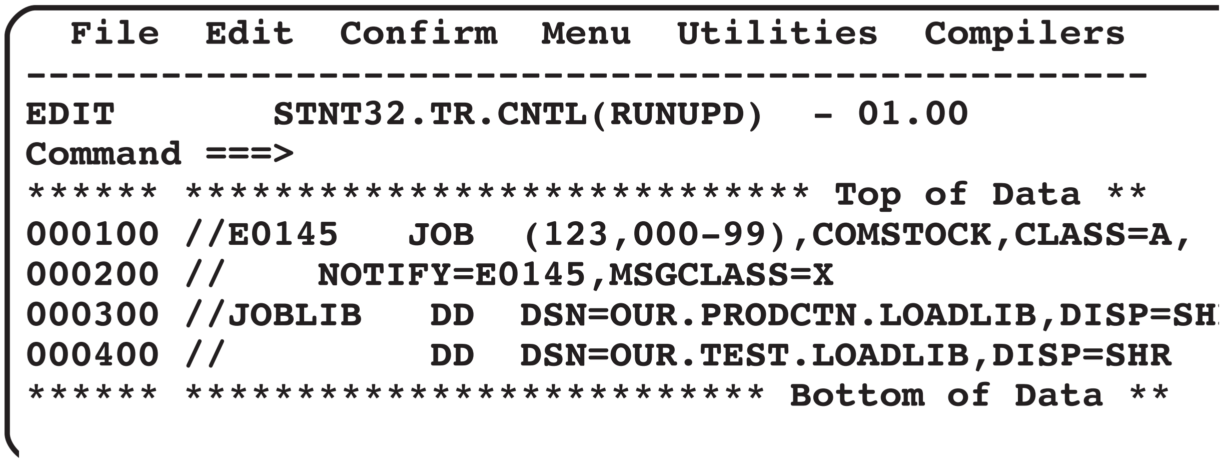
- Then on the command line, issue another copy command, say "copy joblib"

Before



```
  File  Edit  Confirm  Menu  Utilities  Compilers
--------------------------------------------------
EDIT       STNT32.TR.CNTL(RUNUPD)  - 01.00
Command ===> copy joblib
****** ***************************** Top of Data **
000100 //E0145   JOB  (123,000-99),COMSTOCK,CLASS=A,
a00200 //    NOTIFY=E0145,MSGCLASS=X
****** *************************** Bottom of Data **
```

After



```
  File  Edit  Confirm  Menu  Utilities  Compilers
--------------------------------------------------
EDIT       STNT32.TR.CNTL(RUNUPD)  - 01.00
Command ===>
****** ***************************** Top of Data **
000100 //E0145   JOB  (123,000-99),COMSTOCK,CLASS=A,
000200 //    NOTIFY=E0145,MSGCLASS=X
000300 //JOBLIB   DD  DSN=OUR.PRODCTN.LOADLIB,DISP=SH
000400 //         DD  DSN=OUR.TEST.LOADLIB,DISP=SHR
****** *************************** Bottom of Data **
```

JCL Skeletons, continued

- ❐ **At each step, you modify the member you've just brought in to fit the needs of the current JCL you're building**
  - ♦ **Then continue on in this process in the same way until you are done**

- ❐ **This technique can work well in conjunction with the Data Flow Diagrams**

- ❐ **To get you started, we have provided a collection of JCL skeletons which you are free to copy into your library, and then modify to meet your needs**
  - ♦ **The members are currently stored in your TR.CNTL library**
  - ♦ **The members available are**
    - ✗ EXEC
    - ✗ JCLLIB (discussed later)
    - ✗ JOBLIB
    - ✗ NEWDISK
    - ✗ NEWTAPE
    - ✗ OLDDISK
    - ✗ OLDTAPE
    - ✗ OUTPUT (discussed later)
    - ✗ STEPLIB
    - ✗ SYSIN
    - ✗ SYSOUT