z/OS Mainframe Bootcamp

Course Outline

Description

This program is designed as the initial six weeks of training for the Vitality Residency Program which also includes Broadcom's own internal ASE program training as well as specialized product training delivered immediately after this bootcamp is completed.

Audience

Students of this program are new Broadcom Vitality Program candidates who have been recruited and screened by Launchcode.

Prerequisites

There are no prerequisites for this course however previous experience with computers is helpful.

Duration

Thirty days

Course Outline

TSO/ISPF in z/OS

Course Summary

Objectives

By the end of this course, students will be able to:

- LOGON to TSO, get into ISPF/PDF, exit ISPF/PDF, LOGOFF TSO
- Use full-screen terminals, including the appropriate Function keys, to accomplish work under ISPF/PDF, as described below
- Use Workstation terminals, emulating 3270-type terminals or running in GUI mode, f these terminals are available (and, for GUI, if the Client/Server support is installed)
- 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
- Describe the characteristics of, and differences between, sequential data sets, partitioned data sets (PDSs), and PDSEs (Partitioned Data Set Extended)
- View a sequential data set or a member of a PDS/PDSE
- Allocate, rename, and delete data sets or members, and print or display the attributes or contents of a data set
- Copy and move data sets and members
- Use productivity features such as command stacking and split screen processing, the CMDE command and command retrieval techniques
- Edit data sets or members: create new members or files, and modify existing members or files
- Submit batch jobs and work with the output using ISPF 3.8, SDSF, FLASHER, IOF, or (E)JES
- Use ISPF VSAM support to list information about non-VSAM objects and to delete non-VSAM objects
- Use the ISPF Workplace Shell to perform ISPF-based tasks

Topics

- Using Mainframe Computers
- ISPF Introduction
- ISPF Look and Feel
- Working With Data
- Allocating Data Sets
- Looking at Data — Edit, View, and Browse
- More on Edit, View, and Help
- More Utility Functions
- Productivity Tips and Techniques
- EDIT
- Edit and View Primary Commands
- Edit / View — Passing and Receiving Data
- Reference Lists
- Edit Profiles
- Data Set List Utility and Commands
- The ISPF Object / Action Workplace Shell
- Running Batch Jobs
- PDSEs and VSAM Support

Prerequisite

There are no prerequisites for this course, although some experience with programming languages is helpful.

Duration

Three Days

Course Outline

ISPF Update

Course Summary

Objectives

By the end of this course, students will be able to:

- Use the new CUA interface (action bars, pull-downs, point-and-shoot fields, etc.)
- Tailor the look and feel of ISPF to meet individual preferences to accomplish work under ISPF
- 'View' a sequential data set or a member of a PDS/PDSE
- Build and use reference lists
- Use new edit / view commands (e.g.: exclude, flip, shift commands, case changing commands, text split, md, version, level, etc.)
- Use language-sensitive color editing
- Use command retrieval capabilities (e.g.: RETF, RETP) and the TSO command shell retrieval area
- Use ISPF VSAM support to list information about non-VSAM objects and to delete non-VSAM objects
- Manage additional split screens (up to 32) using extensions to the SPLIT and SWAP commands, and the new SCRNAME and SWAPBAR commands
- Use the ISPF Workplace Shell to perform ISPF based tasks
- Use the z/OS UNIX Directory List utility (3.17) to work with z/OS UNIX files.

Topics

- ISPF Update - Introduction
- ISPF Look and Feel
- Utilities and Help
- Looking at Data — Edit, View, and Browse
- More Utility Functions
- Reference Lists
- Edit Commands
- Language-sensitive Color Editing
- Data Set List Utility
- Commands
- The ISPF Command Shell
- Edit commands CUT, PASTE, and Clipboards
- VSAM Support
- The ISPF Object / Action Workplace Shell
- Unix directory list support

Prerequisite

Experience using ISPF is required for this course.

Duration

Two Days

Course Outline

z/OS JCL and Utilities

Course Summary

Objectives

By the end of this course, students will be able to:

- Understand the basic flow of work in z/OS, including JES Readers, Writers, Initiators, the role of the Interpreter, and the purpose of Allocation
- Describe the storage layout of z/OS and use the REGION and MEMLIMIT parameters as appropriate and necessary
- 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
- Create and delete data sets using IEFBR14
- Copy files for backup, restore, and testing purposes using the IBM utility program IEBGENER
- Use some of the basic services of IDCAMS, the VSAM utility
- Use a Sort/Merge program product to sort a sequential data set
- Code the OUTPUT JCL statement to produce multiple groups of SYSOUT files
- Use ISPF/PDF 3.8 and / or SDSF, Flasher, IOF, or (E)JES facilities for tracking jobs and examining job output (as available to the student)
- Code cataloged procedures, including the use of symbolic parameters and defaults, nested procedures, and private proclibs
- Describe the implications of Storage Management Subsystem (SMS) and Partitioned Data Sets, Extended (PDSE's).

Topics

- The Application Program Environment
- Running Jobs
- Introduction to Data Management
- Tape and Disk Data Sets
- Tape and DASD DD Statements
- SMS - System Managed Storage
- Looking at Job output
- Utilities and Job Output Viewing
- Sort / Merge
- OUTPUT Statements
- Condition Code Testing and Memory Management
- JCL procedures
- JCL procedures: inserts and overrides
- Symbolic Parameters
- JCL SETs, INCLUDEs and Nested Procedures
- Additional Data Set Handling Techniques
- TSO Commands
- Sources of Information
- Optional: The COND Parameter

Prerequisites

Some experience with programming languages is helpful, but the main requirement is experience with ISPF (or ROSCOE, etc.).

Duration

Three days

Course Outline

Advanced Topics in z/OS JCL

Course Summary

Objectives

At the end of this course, students will be able to:

- Describe the major components of the z/OS operating system, including JES Readers, Writers, Initiators, role of the Interpreter, the
- purpose of Allocation, and how Work Load Manager tries to tune your system
- Code JCL statements to accomplish work in the z/OS environment
- Code DD statements for multi-volume tape and disk data sets and HFS files
- Debug common JCL-related errors in running jobs
- Create and maintain private libraries (IBM partitioned datasets) using common IBM utility programs: IEFBR14, IEBCOPY
- Copy files for backup, restore, and testing purposes using the IBM utility program IEBGENER
- Use some of the non-VSAM services of IDCAMS, the VSAM utility
- Code OUTPUT JCL statements to produce multiple groups of SYSOUT files
- Code cataloged procedures, including the use of symbolic parameters and defaults
- Create and use Generation Data Groups (GDG)
- Create JCL capable of being step-restarted, if possible
- Describe the implications of Storage Management Subsystem (SMS), including PDSEs (Partitioned Data Set, Extended)
- Use these elements of JCL: IF / THEN, ELSE, ENDIF, SET, INCLUDE, JCLLIB statements, nested procedures, symbolic parameters in open JCL
- Use JES2 or JES3 control statements in the appropriate environment
- Use some advanced features of the Sort, including the use of symbolic names and creating multiple output files.

Topics

- Operating system concepts
- Introduction to Data Management
- Named Data Sets
- Tape DD Statements
- SMS - System Managed Storage
- DASD Concepts
- Condition Code Testing and JCL Debugging
- Special DD Situations
- IBM Utilities
- SORT
- Some Advanced Sort Topics
- OUTPUT Statements
- JCL Procedures
- Additional Techniques
- Some Additional JCL Statement Parameters
- Optional Exercise

Prerequisites

Before taking this course, students should have experience working with JCL.

Duration

Three days

Course Outline

TSO REXX Programming in z/OS

Course Summary

Objectives

By the end of this course, students will be able to:

- Describe the TSO environment, and describe the distinctions between TSO commands and REXX instructions
- Write REXX EXECs to accomplish useful functions
- Use TSO commands to work with datasets, either in native mode or in EXECs
- Use REXX instructions to work with records in files
- Use subroutines as a coding technique for EXECs
- Use TSO and EXECs to run programs in the Foreground or the Background (batch)
- Run EXEC's in the batch, in TSO/E-integrated address spaces or non-TSO/E-integrated address spaces
- Use TSO commands to send and receive datasets between users
- Use the REXX compiler, if it is available
- Use the level 2 REXX constructs, if they are available

Topics

- Introduction
- REXX - Restructured Extended Executor
- RC - Return Code special variable
- REXX PARSEing Capabilities
- Clearing the Screen
- Debugging and TRACE
- SMS - Storage Management Subsystem
- LISTDSI TSO/E Function
- TSO/E EXECIO Command
- Compound Symbols and Stems
- Additional REXX Instructions and Functions
- Error Handling and Condition Traps
- Arithmetic, Conversion, and Boolean Built-in Functions
- Running jobs in the Background
- Host environments and the Dialog Manager

Prerequisite

Experience using ISPF, especially the editor; additionally, experience in submitting jobs to the batch and some programming experience are helpful.

Duration

Five Days

Course Outline

Structured COBOL Workshop for Enterprise COBOL

Course Summary

Objectives

By the end of this course, students will be able to:

- Code and test programs using the "IBM Enterprise COBOL for z/OS and OS/390" compiler to process sequential files
- Describe fields, records, and files to COBOL
- Correctly use the most common COBOL verbs in their various forms
- Use the following techniques in designing or coding COBOL programs: Data editing, including use of multiple currency symbols and the Euro Loop control and switch setting and testing Move mode and locate mode processing Pseudocoding as a design tool Reference modification Some intrinsic functions
- The COBOL COPY statement
- Code COBOL programs using installation standards, with an awareness of the ANSI standard
- Define numeric data items to COBOL that are packed decimal or binary integer in format
- Use the COBOL arithmetic verbs ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPUTE
- Code and test COBOL programs to create reports, including page break processing and control breaks.
- Code and test COBOL programs to perform batch transaction processing using match-merge logic sequential processing of transaction and master files), including update in place for sequential disk files
- Use the following techniques in designing or coding COBOL programs: Top down development, Structured programming, Pseudocoding as a design tool, Modular design
- Code COBOL programs that read from and write to HFS files on systems using z/OS UNIX.

Topics

- Fundamentals
- Describing Data
- Processing Data
- /O Processing Options
- More on Data Items
- PERFORM Statements
- Program Design
- Conditional Statements
- Describing Numeric Data
- Data Alignment
- Arithmetic Instructions
- EVALUATE
- Basic String Manipulation
- Introduction to Intrinsic Functions
- Working with Print Files
- Control Breaks
- Match Merge Logic
- Miscellaneous Topics

Prerequisite

A background in using text editor, experience submitting jobs and examining the output; introduction to data processing concepts are required for this course.

Duration

Five Days

Course Outline

Introduction to z/OS UNIX

Course Summary

Objectives

By the end of this course, students will be able to:

- Accomplish work to support applications that use files in the Hierarchical File System (HFS and zFS), using TSO commands or z/OS UNIX System Services shell commands.
- copy / move data between z/OS files
- copy / move data between HFS files
- copy / move data between z/OS files and HFS files, including converting code pages as necessary
- create, display, rename, delete, archive, unarchive, compress, and uncompress data in HFS files
- edit and browse HFS data
- Use the ISPF shell, ISHELL, to work with HFS files from a TSO/ISPF session
- Display web pages residing in HFS files on the mainframe on his or her local browser on your company intranet
- Create and maintain simple HTML files to display basic text and images and to link from one page to another
- Run batch jobs that access files and programs (DLLs) in the HFS
- Submit jobs to run in the batch from the z/OS UNIX shell
- Use the ISPF UNIX Directory List utiltiy (3.17), if possible
- Use telnet to access z/OS UNIX, if possible in his or her installation
- Understand how UNIX System Services expands and extends the capabilities of the mainframe.

Topics

- Introduction to z/OS UNIX
- More Identities
- Variables
- UNIX Files and the Hierarchical File System (HFS)
- Managing and Deleting Directories and Paths
- Data Transfer - Part 1: TSO Commands
- Displaying Data Under the Shell
- Managing Files and Directories
- Piping and Redirection
- OEDIT and OBROWSE TSO Commands
- ISHELL: Doing UNIX-y Things in an ISPF-y Way
- Locales and Internationalization in UNIX
- More OMVS Features
- Data Transfer - Part 2: UNIX Commands
- Compressing and Uncompressing
- Introduction to The Web
- Introduction to Markup Languages
- Managing Archive Files
- Accessing HFS Files and Programs from Batch and TSO
- Submitting jobs from the shell
- Unix directory list support
- telnet (Optional)
- Wrap Up: Where Do We Go From Here

Prerequisite

Experience working in OS/390 or z/OS; specifically, the student is expected to be comfortable using ISPF and have a clear understanding of MVS-based data organizations is required for this course.

Duration

Three Days

Course Outline

z/OS Assembler Programming Part 1: Beginnings

Course Summary

Objectives

At the end of this course, students will be able to:

- Code a program in Assembler language that uses the following techniques:
  - Use standard save area linkage techniques
  - Define and process sequential files with fixed length records, including
    - Reading and writing records from / to DASD files
    - Reading and writing records from / to tape files
    - Writing records to print files, including formatting detail lines, but not using carriage control characters or other report management techniques
  - Perform calculations using packed decimal arithmetic, including formatting results with edit patterns and half-adjusting results
  - Perform calculations using binary integer arithmetic
  - Work with data in tables, including defining and accessing the elements in a table
  - Use DSECTs to describe structures
  - Use multiple base registers
- Document the program listing with comments to assist in maintenance and understanding of the code
- Debug the resulting code of program-check type errors

Topics

- Fundamentals
- Machine Instruction Formats
- Data Description, Moving Data, Record Processing
- Compares, Branches, and Linkages
- More on Addressability
- Packed Decimal Arithmetic
- More Assembler and Arithmetic Concepts
- Editing Packed Decimal Fields
- A Deeper Look at Instruction Formats
- Binary Integer Data
- More Binary Instructions
- EDMK
- Loops and Tables
- Multiple base registers, DSECTS, ORG
- Working With Bits
- Shift Instructions
- TRT and EX
- Strings
- Setting Addressing Mode

Prerequisites

Before taking this course, students should have a background in using text editor, experience submitting jobs and examining the output; introduction to data processing concepts; ideally some experience in a fourth generation language

Duration

Five days

Course Outline

System Overview

Course Outline

Duration

Three days

Course Outline

Course Outline