Language Environment Overview, Setup and Customization on z/VM

Michael Donovan
CMS Development - IBM Endicott
donovan@us.ibm.com

Prague - 27 April 2001
Session V108

Trademarks

Registered IBM Trademarks

IBM

z/VM

z/OS

VM/ESA

OS/390

DFSMS/MVS

Language Environment

Preface

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Preface (con't)

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environments.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country or not yet announced by IBM. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services.

Front Matter

- Abstract
  - This session will provide a brief overview of the Language Environment (LE) component, a discussion on how applications are affected by its function, an overview of the LE run time options, and how to customize them for maximum efficiency.

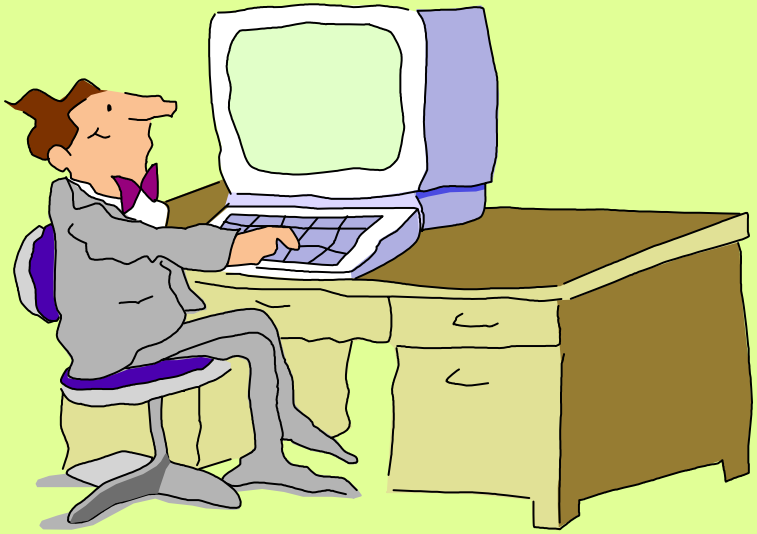
Reflection

- "Language Environment is 100% compatible with previous run time environments" - An unnamed, misinformed marketing rep
- "Read the LE Migration Guide for details of migration problems you **WILL** encounter." - An unnamed LE developer

Agenda

- Why LE?
- Overview/Introduction
- Key Run Time Options
- Setting Run Time Options
- Session Summary
- Sources and Additional Information

Why LE?



- Allow for a consistent means to develop new applications and still be able to utilize legacy programs
- Simplified error reporting
- Consistent storage management and use of other system resources
- etc.

LE Introduction

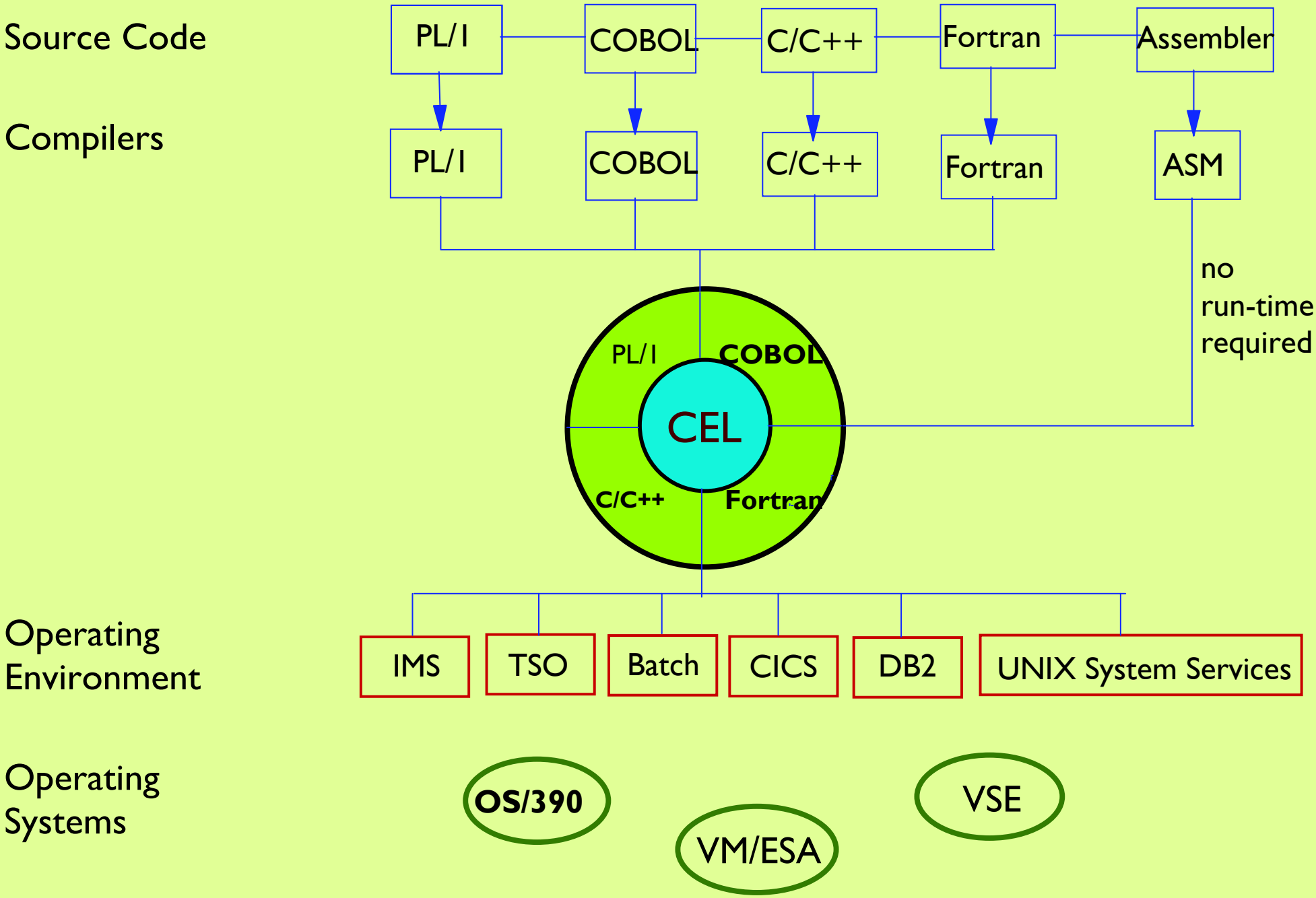
- What is a runtime environment?
  - High level languages encapsulate complicated function into simple "functions".
    - Obtaining storage
    - Obtaining the system date and time
    - Outputting messages
    - Math functions
    - etc.
  - The runtime environment is "called" by the user program to do these pieces of work.
  - LE combines several runtimes.
  - Binding occurs at runtime not link time.

LE Introduction

- Language Environment
  - A common runtime environment across multiple high level languages.
    - COBOL
    - C/C++
    - PL/I
    - FORTRAN (not supported on VM)
    - Assembler (not HLL)
  - Supported across multiple platforms
    - VM (VM/ESA and z/VM)
    - OS/390 and z/OS
    - VSE

LE Introduction

- LE is made up of several components
  - Common Execution Library (CEL) 568819801
  - COBOL Run-Time Library 568819802
  - PL/I Run-Time Library 568819803
  - FORTRAN Run-Time Library 568819804
  - C/C++ Run-Time Library 568819805
  - Visual Age PL/I 568819806



Language Environment for VM,OS/390,VSE

LE Introduction

- CEL is a set of common functions and routines used by all member languages of LE
  - Initialization
  - Storage management
  - Condition handling (includes CEEDUMPs)
  - Message services
  - Date/Time services
  - Math functions
  - Termination

LE Introduction

Supported Releases

LE Release Level	Supported VM/ESA Release Level	Comments
LE 1.5	VM ESA 2.1.0	must be installed
LE 1.6	VM ESA 2.2.0	must be installed
LE 1.8	VM/ESA 2.3.0 VM/ESA 2.4.0 z/VM 3.1.0	preinstalled

NOTE:

LE is upwardly compatible but not downward compatible. Applications linked with a particular release of LE will run on all higher levels of LE (See APAR II11316)

LE Introduction

- Important Libraries
  - Compile Step
    - SCEEMAC MACLIB
  - Prelink step
    - SCEEOBJ TXTLIB
  - Link Edit
    - SCEELKED TXTLIB
  - Run/Go step
    - SCEERUN LOADLIB

LE Introduction

Install Target Minidisks

Minidisk Owner	Default Address	DASD Type	CYLs	Usage
P688198H	2B2	3390	65	Based code shipped with LE
P688198H	2C2	3390	5	Sample files and user modifications
P688198H	2D2	3390	42	Serviced Files
P688198H	2A6	3390	2	AUX files and version vector table that represent your TEST level of LE
P688198H	2A2	3390	2	AUX files and version vector table that represent your PRODUCTION level of LE
P688198H	29E	3390	45	Test build disk. This will normally be copied to MAINT 19E.
P688198H	191	3390	38	P688198H user ID's minidisk. Used for customization.
MAINT	19E	3390	45	Production build disk. MAKE SURE THIS DISK IS LARGE ENOUGH.

Key Run Time Options

- ABTERMENC(option)
  - RETCODE Step ends with return code (job continues)
  - ABEND Step will be ABENDed (job terminates)
  - Recommendations:
    - ABTERMENC(ABEND)

Key Run Time Options

- ALL31(option)
  - OFF For AMODE 24 programs
  - ON For AMODE 31 programs

  - Recommendations:
    - ALL31(ON) whenever possible

Key Run Time Options

- ERRCOUNT(threshhold)
  - >0 The number of conditions allowed before LE terminates the process
  - 0 Disables condition counting
  - Notes:
    - ERRCOUNT(0) required by PL/I for ON-unit processing and C/C++ for signal processing
  - Recommendations:
    - 0

Key Run Time Options

- HEAP(initial, increment, where, type, ..., ...)
  - initial Minimum size of initial heap segment
  - increment Minimum size of add'l segments
  - where BELOW, ANYWHERE
  - type KEEP, FREE action when empty

  - Notes:
    - COBOL working storage
    - Dynamic storage
      (C malloc,PL/I ALLOCATE)

Key Run Time Options

- HEAP(initial, increment, where, type, ..., ...) *(continued)*
  - Recommendations:
    - HEAP(32K,32K,ANY,KEEP,8K,4K)
    - When BELOW use FREE, ANYWHERE use KEEP

Key Run Time Options

- MSGFILE(ddname,....)
  - ddname is the filedef of the dataset to direct runtime diagnostics and reports
  - Recommendations:
    - MSGFILE(SYSOUT,FBA,121,0)

Key Run Time Options

- RPTOPTS(option)
  - ON Provide Options Report upon successful termination
    (currently none if abnormal termination)
  - OFF No Options Report requested

  - Recommendations:
    - RPTOPTS(OFF)

Key Run Time Options

- RPTSTG(option)
  - ON Provide Storage Report upon successful termination
    (none if abnormal termination)
  - OFF No Storage Report requested

  - Notes:
    - Significant performance impact, use for tuning only, never in production
  - Recommendations
    - RPTSTG(OFF)

Key Run Time Options

- STACK(initial, increment, where, type)
  - initial Actual size of initial stack segment
  - increment Minimum size of additional segments
  - where BELOW, ANYWHERE
  - type KEEP, FREE Action when empty
  - Notes:
    - Dynamic Save Area
    - C/C++ and PL/I local variables
    - Must use STACK(,,BELOW) if ALL31(OFF)

Key Run Time Options

- STACK(initial, increment, where, type) *(continued)*
  - Recommendations:
    - STACK(32K,32K,ANY,KEEP)
    - When BELOW use FREE, ANYWHERE use KEEP

Key Run Time Options

- STORAGE(getheap, freeheap, stack, reserve)
  - getheap One byte value used to initialize every heap allocation
  - freeheap One byte value used to initialize every heap free
  - stack One byte value used to initialize every stack allocation
  - reserve Amount of space to reserve for out of storage condition processing

Key Run Time Options

- STORAGE(getheap, freeheap, stack, reserve) *(continued)*
  - Notes:
    - STORAGE(00,,,) is equivalent to COBOL WSCLEAR
    - STORAGE(AA,FE,,) useful for debugging
    - Can be a huge performance impact
  - Recommendations:
    - STORAGE(None, None, None, 8K)

Key Run Time Options

- TERMTHDACT(option)
  - QUIET Messages off
  - MSG Messages only, no dump
  - TRACE Traceback only, no dump
  - DUMP Basic CEEDUMP, no storage
  - UADUMP Full CEEDUMP, system dump
- Recommendation:
  - TERMTHDACT(TRACE)

Key Run Time Options

- TRAP(option)
  - ON,SPIE Condition handling enabled
  - ON,NOSPIE Allows user applications to have their own SPIE routine, LE condition handling will take place via the ESTAE
  - OFF Condition handling disabled - **Avoid**
  - Notes:
    - TRAP(ON) <u>required</u> for LE processing
  - Recommendations:
    - TRAP(ON,SPIE)

Setting Run Time Options

- Installation Level
  - Use the C5688198 EXEC to modify CEEDOPT ASSEMBLE
- Application Level
  - CEEUOPT ASSEMBLE **must** be linked with the application

Setting Run Time Options

- Program Level
  - Compiled into program
    - PLIXOPT for PL/I
    - #pragma runopts for C
    - not available for COBOL
- Program/Command Invocation
  - COBOL (with CBLOPTS(ON))
    - command program args/runtime options
  - C, PL/I, FORTRAN, LE enabled Assembler
    - command runtime options/program args

Summary

- Language Environment provides a common set of runtime functions for HLLs and supports Assembler
- Prior to VM/ESA 2.3, LE had to be manually installed
- Runtime options "customize" LE
  - Use RPTOPTS and RPTSTG to tune your LE
  - TERMTHDACT controls what gets dump on error
  - TRAP determines whether LE does condition handling
  - Options can be customized at different levels

Additional Information

- Program Directory
- LE Programming Reference
- LE Programming Guide
- LE Migration Guide
- All LE documentation available on
  - VM collection CDs
  - LE website
    - http://www.ibm.com/s390/le