z/OS Language Environment Futures Workshop

SHARE In San Francisco, Feb 2013

John Monti
jmonti@us.ibm.com

Copyright IBM Corp. 2013

Disclaimer

- IBM may or may not deliver anything that is discussed in this presentation
- If IBM does deliver something discussed in this presentation it may or may not be in the particular timeframe implied
- Results of this discussion may, in fact, change what and/or when IBM does deliver certain functionality

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- CICS®
- DB2®
- Language Environment®
- OS/390®
- z/OS®

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- A D-APAR is coming soon
- Hey there was a z/OS 2.1 preview yesterday!!!!
- Questions?

D-APAR coming soon!

- PM74657
  - Large AMODE 64 long running servers (like MQ Series) have experienced storage shortages.
  - Obtaining large temporary LE heap storage along with other small requests results in LE heap fragmentation
  - New function “FILL” implemented and available now as ++APARs
    - PTF scheduled to be available by end of February

D-APAR coming soon!

- PM74657
  - Do you need this new functionality?
    - Probably not, unless directed to use it by some component like MQ Series
  - FILL is a new disposition on the HEAP64 run-time options (along with KEEP and FREE) for storage obtained above the bar.
  - FILL works similar to FREE but with a key difference

D-APAR coming soon!

- PM74657
  - FILL
    - Specifies that an increment to user heap storage is release when the last of the storage within that increment is freed. (just like FREE) In addition, when a storage request results in a new increment being created which is greater than the incr64 size, the entire increment will be filled by the single storage request.

z/OS 2.1 preview

- Removal of the ++usermods to create CEEDOPT and CELQDOPT run-time options csects from CEEDOPT, CEECOPT and CELQDOPT samples
  - Statement of direction is part of z/OS V1R12
  - Statement of direction in z/OS V1R13 states that z/OS V1R13 is the last release to support these usermods
  - These usermods will be removed in the release after z/OS V1R13

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- ++USERMOD removal
  - “Installation Default” is renamed to “IBM-Supplied Default”
  - CEEXOPT macro updated to fail when processing CEEDOPT or CELQDOPT with MNOTE 16.
  - CEEDOPT, CEECOPT and CELQDOPT samples removed
  - CEEWDOPT, CEEWCOPT, and CEEWQDOP sample jobs removed
  - New samples CEERDOPT, CEERCOPT and CELQRDOP for CEEROPT generation.

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- ++USERMOD removal
  - Healthcheck "CEE_USING_LE_PARMLIB" is being removed.
    - No longer useful once ++usermods are no longer supported.
  - CEEUOPTs are unaffected!

z/OS 2.1 Preview

- Options report sample

```
Options Report for Enclave main 01/30/13 11:21:31 AM
Language Environment V02 R01.00

LAST WHERE SET                    OPTION
------------------------------------------------------------------
IBM-supplied default                ABPERC(NONE)
IBM-supplied default                ABTERMENC(ABEND)
IBM-supplied default              NOAIXBLD
IBM-supplied default                ALL31(ON)
IBM-supplied default                ANYHEAP(16384,8192,ANYWHERE,FREE)
IBM-supplied default              NOAUTOTASK
Programmer default                  BELOWHEAP(8192,4096,FREE)
```

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- CEEDOPT failure sample

```
<clip>
                                             VCTRSAVE=((OFF),OVR),          X00400000
                                             XUFLOW=((AUTO),OVR)             00420000
** ASMA254I *** MNOTE ***   18+     16,Aborting: CEEDOPT and CELQDOPT processing not
   01-CEEXO
** ASMA254I *** MNOTE ***   19+     16,supported.
   01-CEEXO
                            20                END                            00430000
```

z/OS 2.1 Preview

- HEAP overlay tolerance
  - Request from our L2 organization
  - They see many application errors which cause damaged heap storage
  - Often tricky to debug and NOT an IBM defect.

z/OS 2.1 Preview

- HEAP overlay tolerance
  - Run-time option to request x bytes (8 to 1024) of additional storage be allocated for EACH element in the heap.
- This storage would be set to a value known internally
  - Optionally could be checked when element is freed
    - Significantly cheaper than HEAPCHK
  - Most overlays are small so problems MAY disappear if not checked.
  - Extra storage always multiple of 8 (16 in 64bit)
  - Yes this would apply to HEAPPOOLS and HEAPPOOLS64

z/OS 2.1 Preview

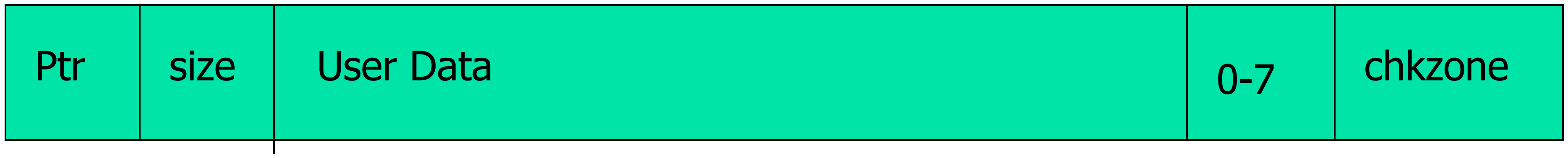
- HEAP overlay tolerance
  - HEAPZONES(size31,output31,size64,output64)
    - Size31 – the amount of storage to add to below the bar (and below the line).
      - Default = 0 (OFF)
    - Output31 – what to do on failures
      - QUIET – Nothing – no checking
      - MSG – Output a message and keep going
      - TRACE – Output a message and a traceback – SHARE!
      - ABEND – ABEND with a U4042 ABEND - default
    - No unique setting for size24/output24

z/OS 2.1 Preview

- HEAP overlay tolerance
  - HEAPZONES(size31,output31,size64, output64)
    - Size64 – the amount of storage to add to above the bar.
      - Default = 0 (OFF)
    - Output64 – what to do on failures
      - QUIET – Nothing – no checking
      - MSG – Output a message and keep going
      - TRACE – Output a message and a traceback – SHARE!
      - ABEND – ABEND with a U4042 ABEND - Default

z/OS 2.1 Preview

- HEAP overlay tolerance
  - RPTSTG will be forced OFF when HEAPZONES is set to ON.
  - We will validate “extra” storage as well.
    - 0-7 bytes of storage not needed
      - 0-15 in AMODE 64
    - User data always rounded up



z/OS 2.1 Preview

- HEAP overlay tolerance
  - HEAPZONES can only be set at the application level
  - HEAPZONES cannot be set at system level or region level
  - HEAPZONES can be set in CLER

z/OS 2.1 Preview

- PAGEFRAMESIZE and PAGEFRAMESIZE64
  - Allows LE storage to be backed by pageable 1M pages.
  - Cannot be set at the system or region level
  - Default continues to back LE storage with 4K pages

z/OS 2.1 Preview

- **PAGEFRAMESIZE**
  - PAGEFRAMESIZE specifies the preferred page frame size in virtual storage for HEAP, ANYHEAP, and STACK storage obtained during application initialization and run-time.

z/OS 2.1 Preview

- **Syntax:**

```
                      .-4K---------------.      .-4K-----------------.
>>--PAGeframesize--(--+------------------+--,--+---------------------+--,->
                      '-heap_frame_size-'      '-anyheap_frame_size-'

                      .-4K---------------.
                  >---+-------------------+--)-->< 
                      '-stack_frame_size-'
```

z/OS 2.1 Preview

- ***heap_frame_size***
  - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments. The page frame size can be specified as one of the following values:
    - **<u>4K</u>** Requests the default 4KB pages.
    - **1M** Requests 1MB pages be used if available.

- ***anyheap_frame_size***
  - Specifies the preferred page frame size in virtual storage for initial anywhere heap storage allocation and any subsequent anywhere heap increments. The page frame size can be specified as one of the following values:
    - **<u>4K</u>** Requests the default 4KB pages.
    - **1M** Requests 1MB pages be used if available.

z/OS 2.1 Preview

- ***stack_frame_size***
  - Specifies the preferred page frame size in virtual storage for initial stack storage allocation and any subsequent stack increments. The page frame size can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests 1MB pages be used if available.

- **Usage Notes:**
  - You cannot set PAGEFRAMESIZE in a CEEPRMxx parmlib member, or with a SETCEE command.
  - You cannot specify PAGEFRAMESIZE with the CEEBXITA assembler user exit interface.
  - In an XPLINK environment, the stack_frame_size suboption only applies to the upward-growing stack.
  - If 1MB page frames are not available, the default 4KB page frame size will be used. No message is issued to indicate this behavior.

z/OS 2.1 Preview

- **Usage Notes:**
  - Page frame sizes larger than 4KB are not allowed below the 16MB line. If a PAGEFRAMESIZE parameter specifies 1MB but that storage type is allocated below the 16MB line, then the default 4KB page frames will be used. No message is issued to indicate this behavior, and the run-time options report will show what the user specified.
  - If any PAGEFRAMESIZE parameter specifies 1M, then all of the storage preallocated to the enclave will request 1MB page frames. The previous two usage notes apply as well.
  - By default, THREADSTACK storage comes from the library heap storage that is allocated with the ANYHEAP run-time option. To use 1MB page frames for the THREADSTACK, ensure the *anyheap_frame_size* suboption specifies 1M.
  - When running in a preinitialized environment with an @GETSTORE service routine, a flag is passed to indicate that storage was requested to be backed by 1MB page frames. For more information about using 1MB page frames with an @GETSTORE service routine, see *z/OS Language Environment Programming Guide*.

z/OS 2.1 Preview

- **PAGEFRAMESIZE64**
  - Similar syntax to PAGEFRAMESIZE but controls AMODE 64 workload
  - **heap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.
  - **heap64_frame_size31**
    - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments above the 16MB line and below the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.

z/OS 2.1 Preview

- **PAGEFRAMESIZE64 (continued)**
  - Similar syntax to PAGEFRAMESIZE but controls AMODE 64 workload
  - **libheap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial library heap storage allocation and any subsequent library heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **<u>4K</u>** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.
  - **libheap64_frame_size31**
    - Specifies the preferred page frame size in virtual storage for initial library heap storage allocation and any subsequent library heap increments above the 16MB line and below the 2GB bar. This value can be specified as one of the following values:
    - **<u>4K</u>** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- **PAGEFRAMESIZE64 (continued)**
  - Similar syntax to PAGEFRAMESIZE but controls AMODE 64 workload
  - **ioheap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial I/O heap storage allocation and any subsequent I/O heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.
  - **ioheap64_frame_size31**
    - Specifies the preferred page frame size in virtual storage for initial I/O heap storage allocation and any subsequent I/O heap increments above the 16MB line and below the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.

z/OS 2.1 Preview

- **PAGEFRAMESIZE64 (continued)**
  - Similar syntax to PAGEFRAMESIZE but controls AMODE 64 workload
  - **stack64_frame_size**
    - Specifies the preferred page frame size in virtual storage for initial stack storage allocation above the 2GB bar.. This value can be specified as one of the following values:
    - **<u>4K</u>** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.

z/OS 2.1 Preview

- **PAGEFRAMESIZE64 (continued)**
- Usage Notes:
  - You cannot set PAGEFRAMESIZE64 during Language Environment installation(CELQDOPT) in a CEEPRMxx parmlib member, or with a SETCEE command, or during region creation(CELQROPT).
  - You cannot specify this option with the CEEBXITA assembler user exit interface.
  - If 1MB page frames are not available, 4KB page frame size will be used. No message is issued to indicate this behavior.
  - Page frame sizes larger than 4KB are not allowed below the 16MB line. If a PAGEFRAMESIZE64 parameter specifies 1MB but that storage type is allocated below the 16MB line, then the default 4KB page frames will be used. No message is issued to indicate this behavior, and the run-time option report will show what the user specified.

z/OS 2.1 Preview

- **PAGEFRAMESIZE64 (continued)**
- Usage Notes:
  - If any PAGEFRAMESIZE64 parameter specifies 1M, then all of the storage preallocated to the enclave will request 1MB page frames. The previous two usage notes apply as well.
  - THREADSTACK64 can get large pages support by CELQPIPI GETSTORE service and SystemLE. If PIPI service routine is in use then CELQPIPI GETSTORE service is used to obtain storage for THREADSTACK64 otherwise SystemLE will allocate storage for THREADSTACK64.
  - When running in a preinitialized environment with a CELQPIPI GETSTORE service routine, a flag is passed to indicate that storage was requested to be backed by 1MB page frames. For more information about using 1MB page frames with a CELQPIPI GETSTORE service routine, see z/OS Language Environment Programming Guide for 64-bit Virtual Addressing Mode.

z/OS 2.1 Preview

- Nested PIPI environments
  - z/OS V1R13 introduced CEEPIPI MAIN_DP environments
    - This support allowed multiple main environments to be initialized
  - z/OS V2R1 extends that support
    - An application running in a MAIN_DP environment can call another application in a nested MAIN_DP environment, without having to return to the assembler PreInit driver to call the second application.

z/OS 2.1 Preview

- To run Preinitialization applications in nested MAIN_DP environments, the following general steps are needed:
  - Create more than one MAIN_DP environment, using the existing CEEPIPI init_main_dp function (integer value = 19), and keep track of the output tokens.
  - Use the existing CEEPIPI call_main function (integer value = 2) with one of the tokens to run application program A.

z/OS 2.1 Preview

- Continued…
  - From application program A, use the CEEPIPI call_main function with another token to invoke application program B in a nested MAIN_DP environment.
  - Application B program can invoke yet another application in a third nested MAIN_DP environment.

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- Nested CEEPIPI support
  - Support allows environments to be created while running in an existing environment.

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- Nested CEEPIPI support
  - Assembler driver: call CEEPIPI(init_main_dp,,,token1)
  - Assembler driver: call CEEPIPI(call_main,,token1,,) to invoke Program_A
  - Program_A: call CEEPIPI(init_main_dp,,,token2) to create a second MAIN_DP environment
  - Program_A: call CEEPIPI(call_main,,token2) to invoke Program_B
  - Program_B: returns to Program_A
  - Program_A: call CEEPIPI(term,token2) to end the second MAIN_DP environment and return to the Assembler driver
  - Assembler driver: call CEEPIPI(term,token1) to end the first MAIN_DP environment.

z/OS 2.1 Preview

- JCL Symbolics
  - Provide the ability for high level languages to gain access to JCL symbolics which have been exported and set.
  - Scheduler component provides a macro, IEFSJSYM, to access these JCL symbolics
    - That interface is assembler!
    - Very powerful
    - Quite complex

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- JCL Symbolics
  - LE callable service
    - Simplifies interface for high level languages
    - CEEGTJS
      - Query a single symbolic per request
      - Value copied into user provided storage
      - Length of value also returned
    - AMODE 64 interface also available
      - __le_ceegtjs()

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- CEEGTJS (function_code, symbol_name, symbol_value, value_length, fc)
  - Function code – 1 (future expansion)
  - Symbol_name – VSTRING
  - Symbol_value – 255 byte fixed-length string (padded with blanks)
  - Symbol_length – length of non-padded symbol
  - Fc – standard LE feedback code
  - NOTE:
    - Lower case characters in the symbol_name will be converted to upper case

z/OS 2.1 Preview

- For 64bit C program
  - #include<__le_api.h>
  - void __le_ceegtjs(_INT4 * function_code,
      - _VSTRING * symbol_name,
      - _CHAR255 * symbol_value,
      - _INT4 * value_length,
      - _FEEDBACK * fc);
  - The parameters are the same as CEEGTJS
    - except fc is a 16-byte feedback code.

z/OS 2.1 Preview

- Some z/OS 2.1 C run-time updates

z/OS 2.1 Preview

- C I/O Performance
  - BSAM type=blocked
    - Support reading, writing, and repositioning of sequential data sets by blocks, rather than by bytes or records
    - Improve the data set operation performance when there is no need to manipulate data within the blocks.
    - Should allow for faster copies and transfers

z/OS 2.1 Preview

- C I/O Performance
  - The support is invoked by calling the fopen()/freopen() functions on a sequential data set with keyword parameter “type=blocked” specified.
    - File must be opened “binary”

z/OS 2.1 Preview

- C I/O Performance
  - Once the data set is opened, other I/O functions can be used to process the stream.
    - fread(), fwrite()
    - rewind(), ftell(), fseek(),ftello() , fseeko(), fgetpos(), fsetpos()
    - fflush() ,fldata() ,fclose()
  - Byte oriented functions are not supported

z/OS 2.1 Preview

- C I/O Performance
  - Buffering
    - For blocked I/O files, buffering is always meaningless.
      - A block is written out as soon as fwrite() completes.
      - The function fflush() has no effect for files opened with “type=blocked”.

z/OS 2.1 Preview

- C I/O Performance
  - Repositioning within files
    - ftell() returns relative block numbers.
      - The behavior of fseek() and ftell() is similar to that when you use relative byte offsets for binary files, except that the unit is a block rather than a byte.
      - For example,
        - fseek(fp,-2,SEEK_CUR);
        - seeks backward two blocks from the current position.
    - You cannot seek past the end or before the beginning of a file.

z/OS 2.1 Preview

- Non-standard function
  - Many other UNIX platforms provide APIs that allow access to select internals of the FILE structure.
  - Assists with the ability to port applications from other UNIX platforms to z/OS.
  - Controlled access to portions of the FILE structure that were previously inaccessible is allowed using new set of APIs.

z/OS 2.1 Preview

- Non-standard function
  - Applications can query an open FILE stream for information about the current status of the stream.
    - Other types of modifiable requests can be performed against the open FILE stream as well.
  - 12 new APIs are defined in a new header called <stdio_ext.h>
  - Non thread-safe versions of 6 of the new APIs are provided
  - The header also defines new macros for use with the APIs

z/OS 2.1 Preview

- Non-standard function
  - size_t __fbufsize(FILE **stream*);
  - int __flbf(FILE **stream*);
  - void _flushlbf(void);   /* yes just 1 underscore */
  - size_t __fpending(FILE **stream*);
  - void __fpurge(FILE **stream*);
  - int __freadable(FILE **stream*);
  - size_t __freadahead(FILE **stream*);
  - int __freading(FILE **stream*);
  - void __fseterr(FILE **stream*);
  - int __fsetlocking(FILE **stream*, int *type*);
  - int __fwritable(FILE **stream*);
  - int __fwriting(FILE **stream*);

z/OS 2.1 Preview

- Non-standard function (unlocked)
  - void _flushlbf_unlocked(void);
  - size_t __fpending_unlocked(FILE **stream*);
  - void __fpurge_unlocked(FILE **stream*);
  - size_t __freadahead_unlocked(FILE **stream*);
  - int __freading_unlocked(FILE **stream*);
  - int __fwriting_unlocked(FILE **stream*);

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- Changes for the latest C standard (C11)
  - Implement four new functions, mbrtoc16(), mbrtoc32(), c16rtomb() and c32rtomb() to convert multibyte characters to and from utf16 and utf32 encoding.

z/OS 2.1 Preview

- Changes for the latest C standard (C11)
  - When a pole error happens, errno will be set to ERANGE. The following functions are modified correspondingly, and are protected by environment variable _EDC_SUSV3=2.
  - log(), logf(), logl()
  - log10(), log10f(), log10l()
  - log1p(), log1pf(), log1pl()
  - log2(), log2f(), log2l()
  - pow(), powl()

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- USS is extending its autoconversion support to include additional CCSIDs
- Customers can no longer assume their files are read or written on the same system, or at the same geographic location from which they originated.
- Tagging and conversion of files between varying code pages is expected to become more prevalent in the future.

Copyright IBM Corp. 2013

z/OS 2.1 Preview

- To enable the conversion environment, a user can use one of the following methods:
  - Set environment variable _BPXK_AUTOCVT to "ALL".
    - setenv() and putenv() have no effect on multi-thread program, non-IPT threads will have the same AUTOCVT state as IPT thread.
  - Using C run-time library function fcntl().
    - fc.cvtcmd = SETCVTALL;
    - fcntl(fd, F_CONTROL_CVT,&fc);

z/OS 2.1 Preview

- New environment variable to set the CCSID for the thread
  - To set a CCSID for a thread.
    - Set environment variable _BPXK_PCCSID to the CCSID.

z/OS 2.1 Preview

- Questions?