Heapzones and Pageframes and Blocks, Oh My!

(or We're Off to See the z/OS 2.1)

John Monti
IBM Poughkeepsie
jmonti@us.ibm.com

SHARE in Boston
August, 2013
Session 13663

Trademarks

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

- CICS®
- IMS
- 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.

IEEE is a trademark in the United States and other countries of the Institute of Electrical and Electronics Engineers, Inc.

POSIX® is a registered Trademark of The IEEE.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, or service names may be trademarks or service marks of others.

* 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.

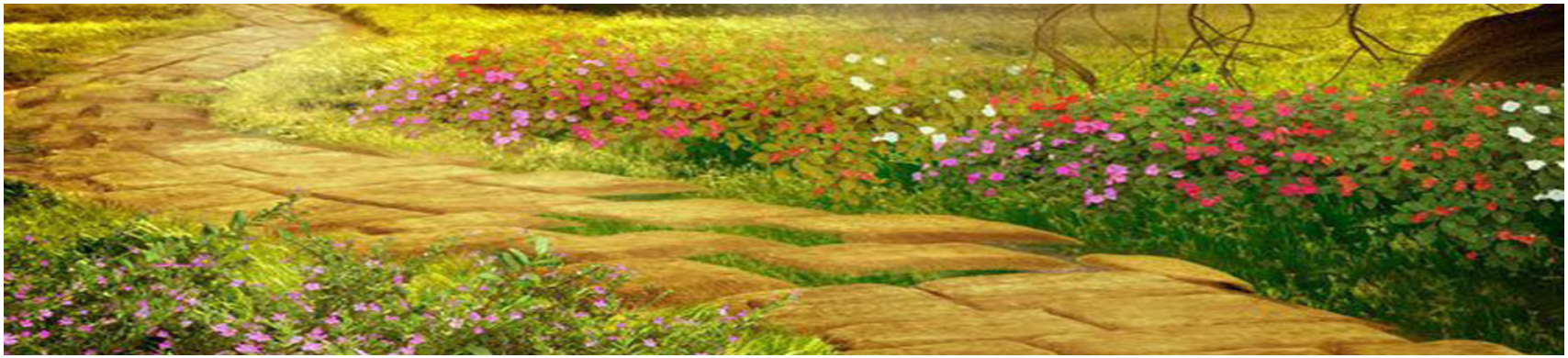
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.

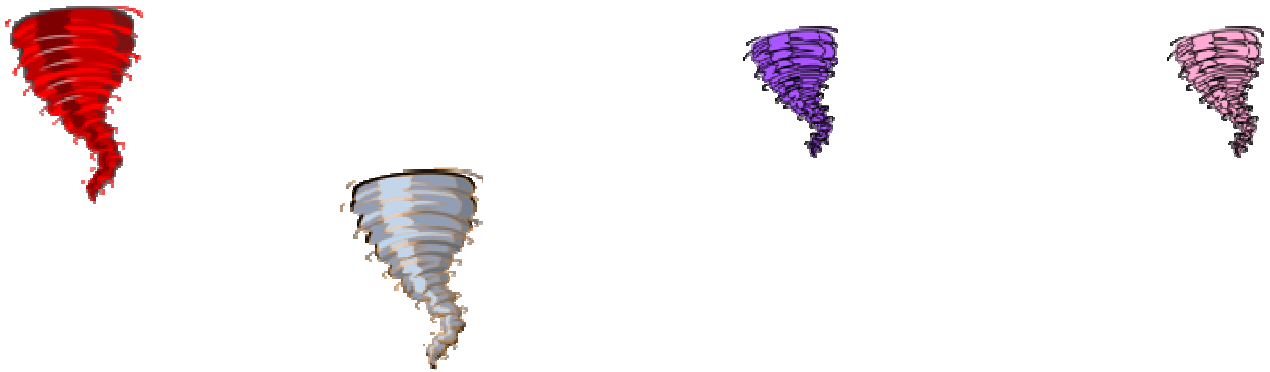
The Yellow Brick Road

- It's a twister! (Changes to the existing landscape)
  - Language Environment D-APARs
- We're off to see the z/OS 2.1
  - Changes for z/OS V2.1 Language Environment
- Ding Dong, the Wicked Witch is Dead!
- Additional information available in Appendix:
  - What's New in z/OS V1.13?



It's a twister!

- Changes to the existing releases through Language Environment D type APARs
  - PM74657 – New "FILL" value on STORAGE Run-time Option
  - PM87183 – Language Environment support for COBOL V5.1
  - PM92330 – Additional Language Environment support for COBOL V5.1
  - PM94340 – Loop after application overlays PICB control block

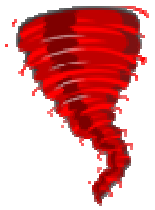


It's a twister!

- PM74657
  - Large, long running AMODE 64 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
  - PTFs
    - Release 770 (R12) : UK91509
    - Release 780 (R13) : UK91510
    - In base of V2.1

It's a twister!

- PM74657
  - 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
  - FILL
    - Specifies that an increment to user heap storage is released 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.



It's a twister!

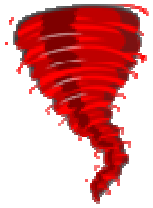
- PM74657
  - Do you need this new functionality?
    - Probably not, unless directed to use it by some component like MQ Series

It's a twister!

- PM87183
  - Language Environment support for COBOL V5.1
  - Required changes to base Language Environment to support the new COBOL compiler and new Language Environment COBOL runtime.
  - Additional PTFs required from COBOL and other related products
  - PTFs
    - Release 780 (R1.13) : UK94564
    - Release 790 (R2.1) : UK94565

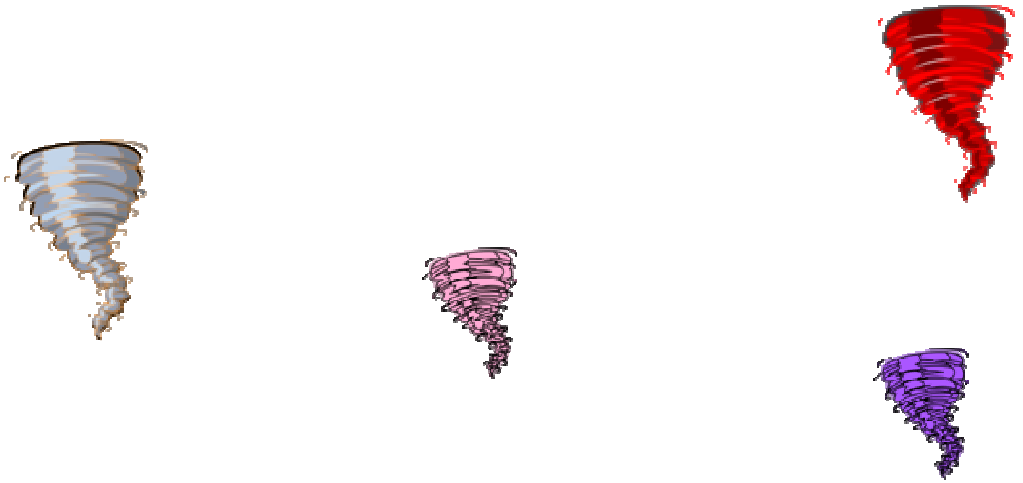
It's a twister!

- PM92330
  - Additional Language Environment support for COBOL V5.1
  - The new V5.1 COBOL manages WORKING-STORAGE differently than the previous versions.
  - Customers have become accustomed to using the Language Environment Storage Report to tune their applications' WORKING-STORAGE usage.
  - This change will allow the V5.1 COBOL WORKING-STORAGE to again be allocated from Language Environment heap storage and therefore tuned with the storage report.




It's a twister!

- PM92330
  - This support is not currently available but is planned to be delivered before the end of 2013.
  - Any tuning you may be doing with V5.1 COBOL should take this into account as there will be no mechanism to "turn off" this change.



It's a twister!

- PM94340
  - Loop after application overlays PICB control block under CICS
  - SUG APAR PM40489 taken previously with intent of code being delivered in a new release in 2012.
    - However, there was no release in 2012… So rolling this support to R13

We're off to see the z/OS 2.1

- Changes in Language Environment for z/OS V2.1
  - HEAPZONES
  - PAGEFRAMESIZE and PAGEFRAMESIZE64
  - C/C++ fopen() type=blocked
  - Nested CEEPIPI environments
  - JCL Symbolics
  - C non-standard I/O functions
  - C11 standards changes
  - Auto-conversion enhancements
  - Runtime Option ++USERMOD Removal

We're off to see the z/OS 2.1

- HEAPZONES
  - 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.

We're off to see the z/OS 2.1

- HEAPZONES
  - 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

We're off to see the z/OS 2.1

- HEAPZONES
  - 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

We're off to see the z/OS 2.1

- HEAPZONES
  - 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

We're off to see the z/OS 2.1

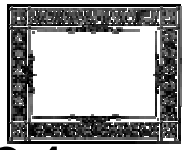
- HEAPZONES
  - HEAP overlay tolerance
    - RPTSTG will be forced OFF when HEAPZONES is set to ON.
    - We will validate "extra" storage as well.
      - 0-7 bytes of storage not needed
        - 0-15 in AMODE 64
      - User data always rounded up

Ptr	size	User Data	0-7	chkzone

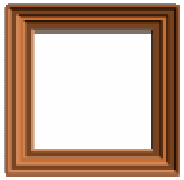
We're off to see the z/OS 2.1

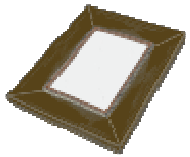
- HEAPZONES
  - HEAP overlay tolerance
    - HEAPZONES can only be set at the application level
    - HEAPZONES cannot be set at system level or region level
    - HEAPZONES can be set in CLER

We're off to see the z/OS 2.1

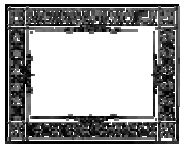


- PAGEFRAMESIZE and PAGEFRAMESIZE64
  - Allows Language Environment storage to be backed by pageable 1M pages.
  - Cannot be set at the system or region level
  - Default continues to back LE storage with 4K pages
  - PAGEFRAMESIZE was available on R13 but not initially supported until the EC12 GAed.





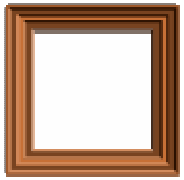
We're off to see the z/OS 2.1

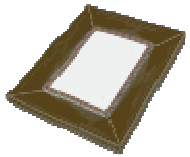


- **PAGEFRAMESIZE (R13)**
  - PAGEFRAMESIZE specifies the preferred page frame size in virtual storage for HEAP, ANYHEAP, and STACK storage obtained during application initialization and runtime.
- **Syntax:**

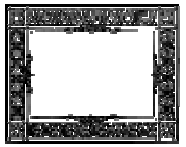
```
                          .-4K--------------.     .-4K-----------------.
>>--PAGeframesize--(--+-----------------+--,--+-------------------+--,->
                      '-heap_frame_size-'     '-anyheap_frame_size-'

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

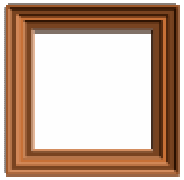


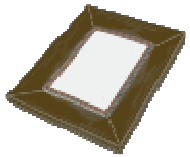


We're off to see the z/OS 2.1

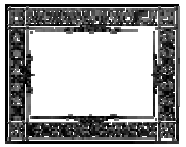


- ***heap_frame_size***
  - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments. The page frame size can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests 1MB pages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

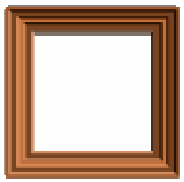


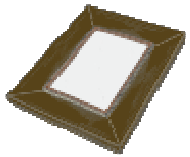


We're off to see the z/OS 2.1

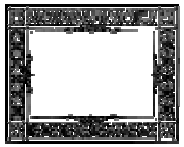


- ***stack_frame_size***
  - Specifies the preferred page frame size in virtual storage for initial stack storage allocation and any subsequent stack increments. The page frame size can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests 1MB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

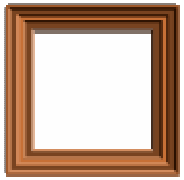


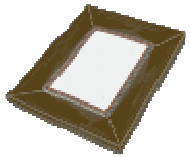


We're off to see the z/OS 2.1

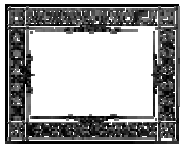


- **Usage Notes:**
  - Page frame sizes larger than 4KB are not allowed below the 16MB line. If a PAGEFRAMESIZE parameter specifies 1MB but that storage type is allocated below the 16MB line, then the default 4KB page frames will be used. No message is issued to indicate this behavior, and the run-time 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*.

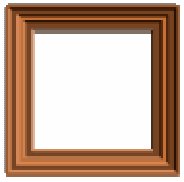


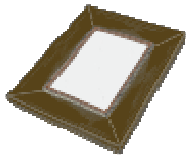


We're off to see the z/OS 2.1

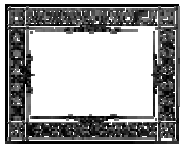


- **PAGEFRAMESIZE64**
  - Similar syntax to PAGEFRAMESIZE but controls AMODE 64 storage management
  - **heap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments above the 2GB bar. This value can be specified 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.

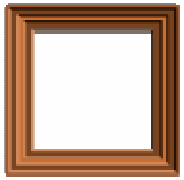


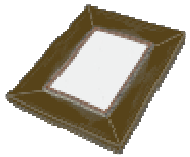


We're off to see the z/OS 2.1

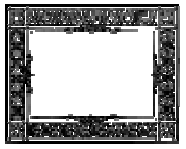


- **PAGEFRAMESIZE64**
  - **heap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial heap storage allocation and any subsequent heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB 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.

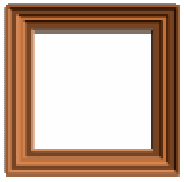


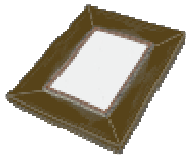


We're off to see the z/OS 2.1

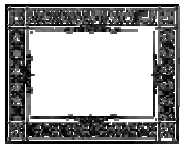


- **PAGEFRAMESIZE64 (continued)**
  - **libheap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial library heap storage allocation and any subsequent library heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **4K** 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:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests the 1MB pages to be used if available.

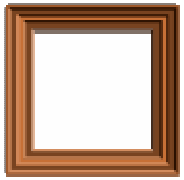


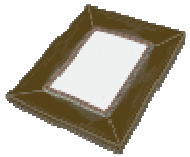


We're off to see the z/OS 2.1

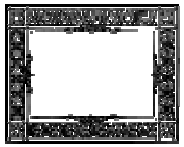


- **PAGEFRAMESIZE64 (continued)**
  - **ioheap64_frame_size64**
    - Specifies the preferred page frame size in virtual storage for initial I/O heap storage allocation and any subsequent I/O heap increments above the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests 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.

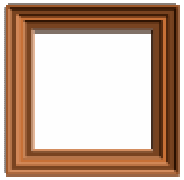


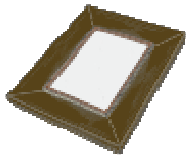


We're off to see the z/OS 2.1

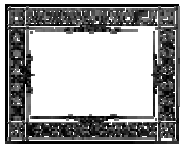


- **PAGEFRAMESIZE64 (continued)**
  - **stack64_frame_size**
    - Specifies the preferred page frame size in virtual storage for initial stack storage allocation above the 2GB bar. This value can be specified as one of the following values:
    - **4K** Requests the default 4KB pages.
    - **1M** Requests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

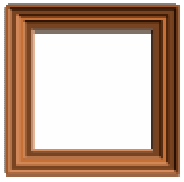


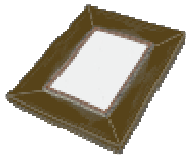


We're off to see the z/OS 2.1

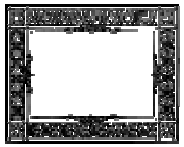


- **PAGEFRAMESIZE64 (continued)**
- Usage Notes:
  - You cannot set PAGEFRAMESIZE64 in a CEEPRMxx parmlib member, or with a SETCEE command, or during region creation(CELQROPT).
  - You cannot specify this option with the CEEBXITA assembler user exit interface.
  - If 1MB page frames are not available, 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.

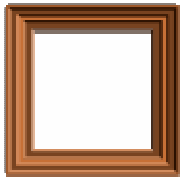


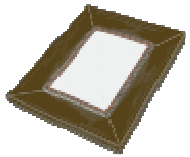


We're off to see the z/OS 2.1



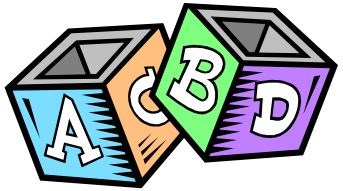
- **PAGEFRAMESIZE64 (continued)**
- Usage Notes:
  - If any PAGEFRAMESIZE64 parameter specifies 1M, then all of the storage preallocated to the enclave will request 1MB page frames. The previous two usage notes apply as well.





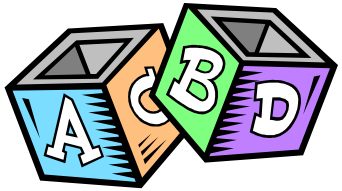
We're off to see the z/OS 2.1

- **Blocks – Oh my!**
  - BSAM type=blocked for C/C++ fopen()
    - 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
    - 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"



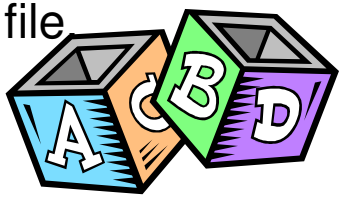
We're off to see the z/OS 2.1

- **Blocks – Oh my!**
  - 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
  - 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".

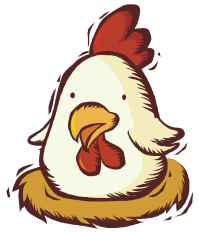


We're off to see the z/OS 2.1

- **Blocks – Oh my!**
  - Repositioning within files
    - ftell() returns relative block numbers.
      - The behavior of fseek() and ftell() is similar to 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.

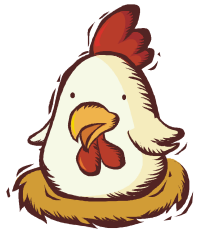


We're off to see the z/OS 2.1



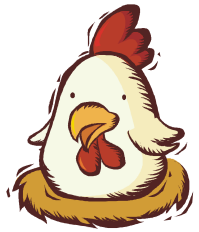
- **NESTED CEEPIPI MAIN ENVIRONMENTS**
  - z/OS V1R13 introduced CEEPIPI MAIN_DP environments
    - This support allowed multiple main environments to be intialized
  - z/OS V2R1 extends that support
    - An application running in a MAIN_DP environment can call another application in a nested MAIN_DP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

We're off to see the z/OS 2.1



- **NESTED CEEPIPI MAIN ENVIRONMENTS**
  - 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.
    - 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.

We're off to see the z/OS 2.1



- **NESTED CEEPIPI MAIN ENVIRONMENTS**
  - Support also allows environments to be created while running in an existing environment.
    - 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.

We're off to see the z/OS 2.1

- **JCL Symbolics**
  - Provide the ability for high level languages to gain access to JCL symbolics which have been exported and set.
  - Scheduler component provided assembler macro IEFSJSYM to access these JCL symbolics
    - That interface is assembler
    - Very powerful
    - Quite complex
  - LE callable service to simplify the 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()

We're off to see the z/OS 2.1

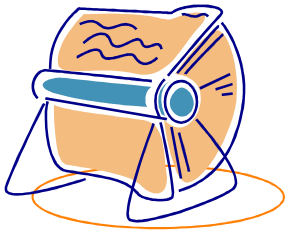
- **JCL Symbolics**
  - CEEGTJS (function_code, symbol_name, symbol_value, value_length, fc)
    - function_code – 1 (future expansion)
    - symbol_name – VSTRING
    - symbol_value – 255 byte fixed-length string (padded with blanks)
    - symbol_length – length of non-padded symbol
    - fc – standard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

We're off to see the z/OS 2.1

- **JCL Symbolics**
  - 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, of course, fc is a 16-byte feedback code.

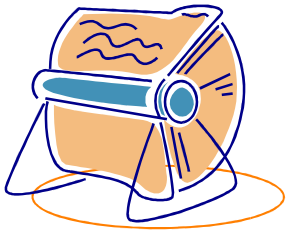
We're off to see the z/OS 2.1

- **Non-standard C/C++ I/O functions**
  - 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.
  - 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



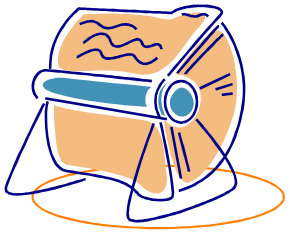
We're off to see the z/OS 2.1

- **Non-standard C/C++ I/O functions**
  - 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_);



We're off to see the z/OS 2.1

- **Non-standard C/C++ I/O functions**
  - **Unlocked versions:**
    - 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*);



We're off to see the z/OS 2.1

- **C11 standard updates**
  - 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.
    - 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()

We're off to see the z/OS 2.1

- **Unix System Services Auto-conversion Updates**
  - USS is extending its autoconversion support to in 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.
  - 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);

We're off to see the z/OS 2.1

- **Unix System Services Auto-conversion Updates**
  - 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.

Ding Dong, the wicked witch is dead!

- Run-time option ++USERMODs removed
  - 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
    - Its now z/OS V2R1:
      - The run-time option ++USERMODs are NO LONGER SUPPORTED!

Ding Dong, the wicked witch is dead!

- Run-time option ++USERMODs removed
  - "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.

Ding Dong, the wicked witch is dead!

- Run-time option ++USERMODs removed
  - Healthcheck "CEE_USING_LE_PARMLIB" is being removed.
    - No longer useful once ++USERMODs are no longer supported.
  - CEEUOPTs are unaffected!

Ding Dong, the wicked witch is dead!

- Run-time option ++USERMODs removed
- 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)
```

Ding Dong, the wicked witch is dead!

- Run-time option ++USERMODs removed
- 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
```

The End..

Thank you!

Appendix

- What's New in z/OS V1.13?
- Sources for Additional Information

High Register Support

- PL/I and C/C++ compilers are capable of exploiting high halves of 64-bit general purpose registers in AMODE 31 applications
  - Integer arithmetic operations
  - Instructions introduced by the High Word Facility
- Language Environment has been enhanced:
  - to ensure this usage does not impact callers
  - to provide full register contents for diagnostic purposes

High Register Support

- Support to save/restore full 64-bit register contents when AMODE 31 application is called from the operating system or by another driver program
- AMODE 31 CEEDUMP support to display full 64 bit registers
  - When unavailable the high half of the 64 bit register is displayed as '********'
- AMODE 31 LEDATA support to display high halves of 64 bit registers when formatting the MCH control block
- CEEDUMP/LEDATA support rolled back via PM04026
  - V1R10: UK59090 V1R11: UK59091

High Register Support

- **PM04026 – High Register Support**
  - **CEEDUMP**

```
Machine State:

      ILC..... 0002    Interruption Code..... 0009

      PSW..... 078D2400 A19C60FE

      GPR0..... 00000000_00000000  GPR1..... 00000000_0000000A  GPR2.....
   00000000_A1CD09BC  GPR3..... 00000000_219C60B8

      GPR4..... 00000000_2199D2D8  GPR5..... 00000000_21F91A00  GPR6.....
   00000000_21F92AC8  GPR7..... 00000000_219BDE40

      GPR8..... 00000000_A19C63A8  GPR9..... 00000000_21F93368  GPR10....
   00000000_A19C6070  GPR11.... 00000000_A19C60A0

      GPR12.... 00000000_21713B58  GPR13.... 00000000_2199D6D8  GPR14....
   00000000_00000000  GPR15.... 00000000_00000006
```

High Register Support

- **PM04026 – High Register Support**
  - **IPCS**

```
Machine State
+000248  MCH_EYE:ZMCH
+000250  GPR00:00000000    GPR01:0000000A
+000258  GPR02:A1CD09BC    GPR03:219C60B8
+000260  GPR04:2199D2D8    GPR05:21F91A00
+000268  GPR06:21F92AC8    GPR07:219BDE40
+000270  GPR08:A19C63A8    GPR09:21F93368
+000278  GPR10:A19C6070    GPR11:A19C60A0
+000280  GPR12:21713B58    GPR13:2199D6D8
+000288  GPR14:00000000    GPR15:00000006
+000290  PSW:078D2400 A19C60FE
```

High Register Support

- **PM04026 – High Register Support**
  - **IPCS**

```
+000388  GPR_H00:00000000        GPR_H01:00000000
+000390  GPR_H02:00000000        GPR_H03:00000000
+000398  GPR_H04:00000000        GPR_H05:00000000
+0003A0  GPR_H06:00000000        GPR_H07:00000000
+0003A8  GPR_H08:00000000        GPR_H09:00000000
+0003B0  GPR_H10:00000000        GPR_H11:00000000
+0003B8  GPR_H12:00000000        GPR_H13:00000000
+0003C0  GPR_H14:00000000        GPR_H15:00000000
```

CEEPIPI Multiple Main Support

- Additional interfaces provided in Preinit to facilitate conversion from the Preinitialization Compatibility Interface (PICI) to Preinit / CEEPIPI
  - Support for multiple main environments on one TCB;
  - Support for a user word that can be accessed from both outside and within a Preinit environment

CEEPIPI Multiple Main Support

Support for Multiple Main Environments on one TCB

- New CEEPIPI function: init_main_dp
- Allows the Preinit assembler driver to create multiple main CEEPIPI environments on the same TCB
- Main programs can be called on these environments, but only one call can be active at a time on a given TCB

CEEPIPI Multiple Main Support

Support for Multiple Main Environments on one TCB...

CALL CEEPIPI(init_main_dp,ceexptbl_addr,service_rtns,token)

- init_main_dp (input) - A fullword containing the init_main_dp function code (integer value = 19).
- ceexptbl_addr (input) - A fullword containing the address of the PreInit table to be used during initialization of the new environment.
- service_rtns (input) - A fullword containing the address of the service routine vector or 0, if there is no service routine vector.
- token (output) - A fullword containing a unique value used to represent the environment.

CEEPIPI Multiple Main Support

Support for Preinit User Word

- Facilitates communication between the Preinit assembler driver and the user code running within a Preinit environment

- Preinit assembler driver uses CEEPIPI interfaces to access the user word
  - CEEPIPI(set_user_word,...) sets the user word value
  - CEEPIPI(get_user_word,...) retrieves the user word value from the last set_user_word call

CEEPIPI Multiple Main Support

Support for Preinit User Word…

- Code running within Preinit environment accesses the user word from within the CAA control block
  - Field CEECAA_USER_WORD in the assembler CEECAA mapping
    - 4 byte field located at offset +3F0x
  - Modifications to this field by the user code running in the Preinit environment are not saved between CEEPIPI calls
    - Next CEEPIPI call will use value from last set_user_word call

CEEPIPI Multiple Main Support

Support for Preinit User Word…

CALL CEEPIPI(set_user_word,token,value)

- set_user_word (input) - A fullword containing the set_user_word function code (integer value = 17)
- token (input) - A fullword with the value of the token of the environment
- value (input) - A fullword value that will be used to initialize the user word in the initial thread CAA when the application is invoked

CEEPIPI Multiple Main Support

Support for Preinit User Word…

CALL CEEPIPI(get_user_word,token,value)

- get_user_word (input) - A fullword containing the get_user_word function code (integer value = 18)
- token (input) - A fullword with the value of the token of the environment
- value (output) - A fullword that will be returned containing the current value that will be used to initialize the CAA user word when the next application is invoked

Deferred Debug Support

- Support to allow debugging to start at a particular C/C++ entry point
  - Currently available for COBOL and PL/I
- New callable service CEEKRGPM provided
- New RCB fields
  - CEERCB_PMUSER – pm_user address supplied on the CEEKRGPM call
  - CEERCB_PMADDR – pm_addr address supplied on the CEEKRGPM call
- Support available via PTFs for APAR PM15192
  - V1R10: UK90027 V1R11: UK90028 V1R12: UK90029

Deferred Debug Support

- CEEKRGPM callable service

```
+--- Syntax ----------------------------------------------+
|                                                         |
|  CEEKRGPM ( pm_addr, rsvd_word1, pm_user, [fc] )        |
|     POINTER    *pm_addr;                                |
|     INT4       *rsvd_word1;                             |
|     POINTER    *pm_user;                                |
|     FEED_BACK  *fc;                                     |
|                                                         |
+---------------------------------------------------------+
```

- Call this CWI interface as follows:

```
L    R15,CEECAACELV-CEECAA(,R12)
L    R15,68(,R15)
BALR R14,R15
```

Deferred Debug Support

- CEEKRGPM callable service
  - pm_addr (input)
    - The address of a pattern match routine that is to be registered or zero when no pattern match routine should be registered (de-registration).
  - rsvd_word1 (input)
    - A fullword reserved for future use. This must be set to zero.
  - pm_user (input)
    - The address of an area supplied by the user for the user's use, or zero when no user area is provided. (The user is responsible for freeing this area, if necessary.)
  - fc (output/optional)

Deferred Debug Support

- CEEPIPI call_sub
  - If CEERCB_PMADDR is non-zero
    - The pattern match routine will be called specifying function code 177, along with the name and entry point (from the PIPI table) of the call_sub routine about to be invoked, as well as the supplied pm_user address.
    - This call will occur just prior to calling the debug event handler with PIPI Subroutine Initialization event (115)
    - The pattern match routine will not be called for call_sub_addr, call_sub_addr_nochk or call_sub_addr_nochk2 requests
    - The pattern match routine will not be called if no routine name is available in the PIPI table

DCE Removal

- Starting in z/OS V1R13, the z/OS Distributed Computing Environment (DCE) and Distributed Computing Environment Security Server (DCE Security Server) will no longer ship with z/OS.
- Minor updates made to documentation and C headers

DCE Removal

- pthread.h
  - Removed inclusion of <dce/dce_pthreads.h>
  - Added a #error message in place of the header inclusion to gracefully fail the compile with a meaningful message.
  - This header also contains equates __COND_DCE and __MUTEX_DCE. There are no references in the external publications for these definitions. Definitions for these have been left in the header for compatibility.

DCE Removal

- signal.h
  - Removed inclusion of <dce/dce_signal.h>
  - Added a #error message in place of the header inclusion to gracefully fail the compile with a meaningful message.
  - This header also contains a signal definition called SIGDCE. References to this signal have been removed or reworded in the external publications, but signal definition in the header has been left for compatibility.

DCE Removal

- Update CEE5224W explanation

CEE5224W The signal SIGDCE was received.

Explanation: **DCE has been removed as of z/OS V1R13. On previous releases** the SIGDCE signal was generated as a result of a MODIFY DCEKERN,DEBUG pid= command. It communicates to a DCE-enabled process a desire to enable DCE run-time debug messages. If the target process is not a DCE process, the target process does not know how to handle SIGDCE.

Programmer response: None.

System action: No system action taken.

Symbolic feedback code: CEE538

I/O Abend Recovery

- The C-RTL is unable to ignore certain DFSMS abends, requiring application developers to write condition handlers or SIGABND handlers to attempt recovery.
- I/O Abend Recovery provides a mechanism to enable the C-RTL to recover gracefully from an abend condition during output or CLOSE processing, when the abend can not be ignored.
- The C-RTL function that triggered the abend condition will gracefully return to the application instead of bringing down the enclave (if condition handling is not in effect).

I/O Abend Recovery

- Two different ways to invoke abend recovery behavior:
  - New fopen()/freopen() keyword
  - New environment variable
- Either method can be used to control how the C-RTL treats abend conditions that can not be ignored.
- When either method is set up to recover from the abend, the C-RTL will instruct the function to return a failing value to the application and set errno to 92.
- Diagnostic information will also be set in the __amrc structure.

I/O Abend Recovery

- fopen()/freopen() keyword usage
  - abend=**abend** | recover
  - **abend** instructs the runtime library to ignore abend conditions that can be ignored. No attempt is made to recover from abend conditions that cannot be ignored.
  - **recover** instructs the runtime library to attempt to recover from an abend issued during certain low-level I/O operations (WRITE / CHECK sequence and CLOSE).
  - This method specifies the behavior for only the stream being opened.
  - This method overrides the setting of the _EDC_IO_ABEND environment variable

```
fopen("//'myfile.data'","wb,type=record,abend=recover");
```

I/O Abend Recovery

- Environment variable usage
  - Environment variable name is _EDC_IO_ABEND
  - **ABEND** and RECOVER are the possible values
    - The values invoke the same behavior as their relative fopen() keyword values.
    - When unset or set to something other than RECOVER, the default behavior is ABEND.
    - The setting of the environment variable defines the behavior for the life of an open stream
    - The environment variable can be overridden by the fopen() keyword.

```
setenv("_EDC_IO_ABEND", "RECOVER", 1);
```

I/O Abend Recovery

- CEECAA Updates (with offsets)
  - **CEECAASHAB_RECOVER_IN_ESTAE_MODE (+30C)**
    - [Bit in the CEECAAFLAG1 field] When ON, the Language Environment ESTAE resumes to the abend shunt in the mode and key in which the Language Environment ESTAE was established.
  - **CEECAASHAB_KEY (+30D)**
    - [Character] IPK result when CEECAASHAB is set.
  - These fields are added to safe guard against key switching and doing a retry in a different key than which the recovery routine was established.

BSAM Greater Than 64K Tracks

- XL C/C++ Run-time Library support for large format sequential data sets greater than 65,535 tracks/volume when opened for BSAM (seek) under binary and text I/O
  - This is Part III of the C/C++ RTL efforts to exploit DFSMSdfp support for large format sequential data sets
    - V1R8 – data sets opened for QSAM I/O
    - V1R12 – data sets opened for BSAM (seek) under record I/O
  - Note: Allocation of a new large format sequential data set can be accomplished by specifying the keyword DSNTYPE=LARGE on a JCL DD statement or using the dynamic allocation equivalent.
  - Supported data set types: large format sequential data sets that are single or multivolume, residing on SMS-managed or non-SMS managed devices, catalogued or uncatalogued.

BSAM Greater Than 64K Tracks

- XL C/C++ Run-time Library support for large format sequential data sets…
  - Invoked by calling the fopen() function on a pre-existing large format sequential data set (DNSTYPE=LARGE was specified).
  - New macro __DSNT_LARGE added for use with dynalloc() function.
  - No support for the fopen() or freopen() equivalent of DSNTYPE=LARGE
  - Once the data set is opened, other OS I/O functions can be used to process the stream.

BSAM Greater Than 64K Tracks

- Large Files versions of ftello() and fseeko() have been added that work on large format sequential data sets that are opened for any type of I/O (record, binary or text)
  - Allows reporting of and repositioning, either directly or relatively, to offsets greater than 2GB – 1.
    - AMODE 31 ftell() and fseek() behavior remains unchanged: offsets used for reporting and repositioning are still limited to 2GB - 1.
  - In order to use the large files versions of ftello() and fseeko(), define the following feature test macro in your application:

```
#define _LARGE_FILES 1
```

    - fgetpos() and fsetpos() can be used with large files without any source code modification

Sources for Additional Information

- Language Environment Debugging Guide
- Language Environment Run-Time Messages
- Language Environment Programming Reference
- Language Environment Programming Guide
- Language Environment Programming Guide for 64-bit Virtual Addressing Mode
- Language Environment Customization
- Language Environment Run-Time Application Migration Guide
- Language Environment Writing ILC Applications
- Language Environment Vendor Interfaces
- Language Environment Concepts Guide
- MVS IPCS Commands
- CICS Supplied Transactions