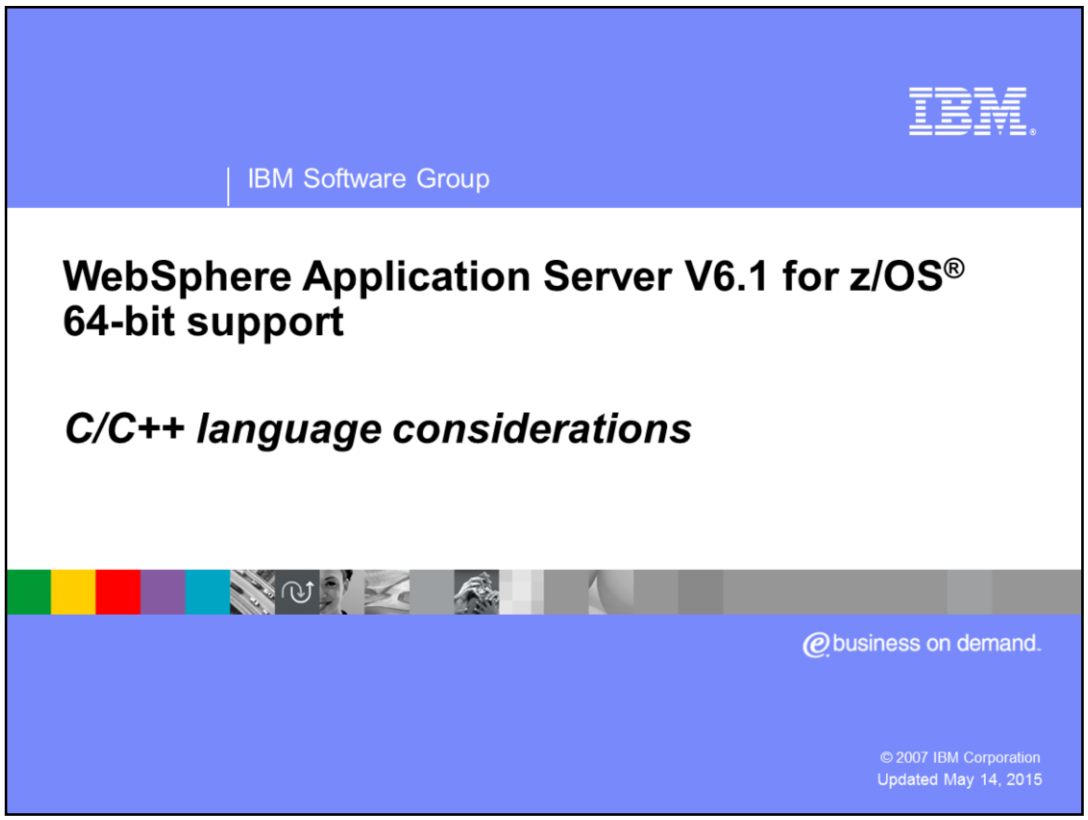
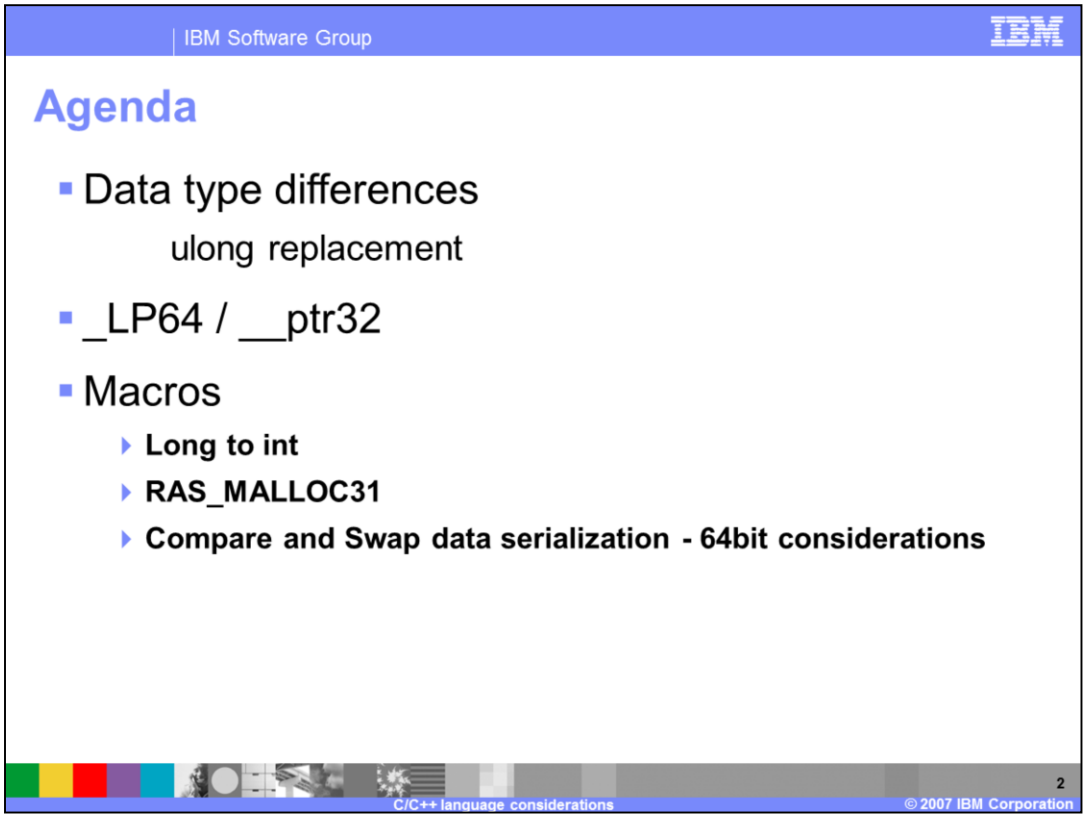
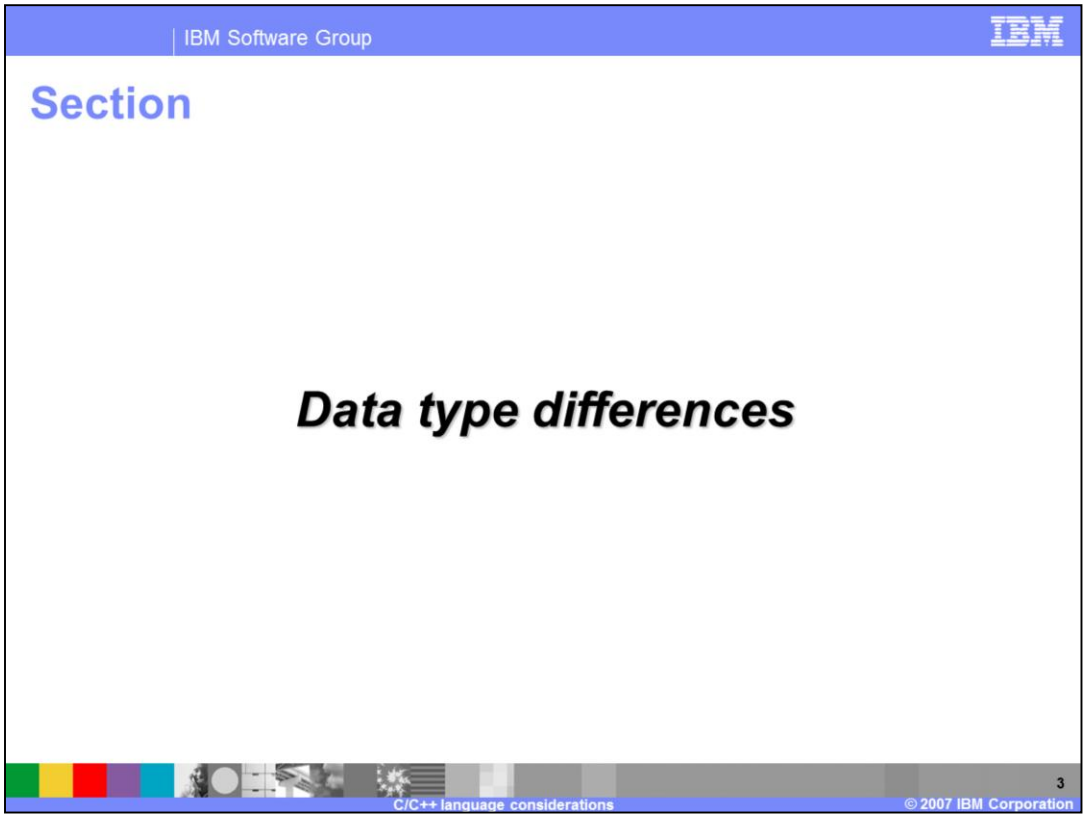


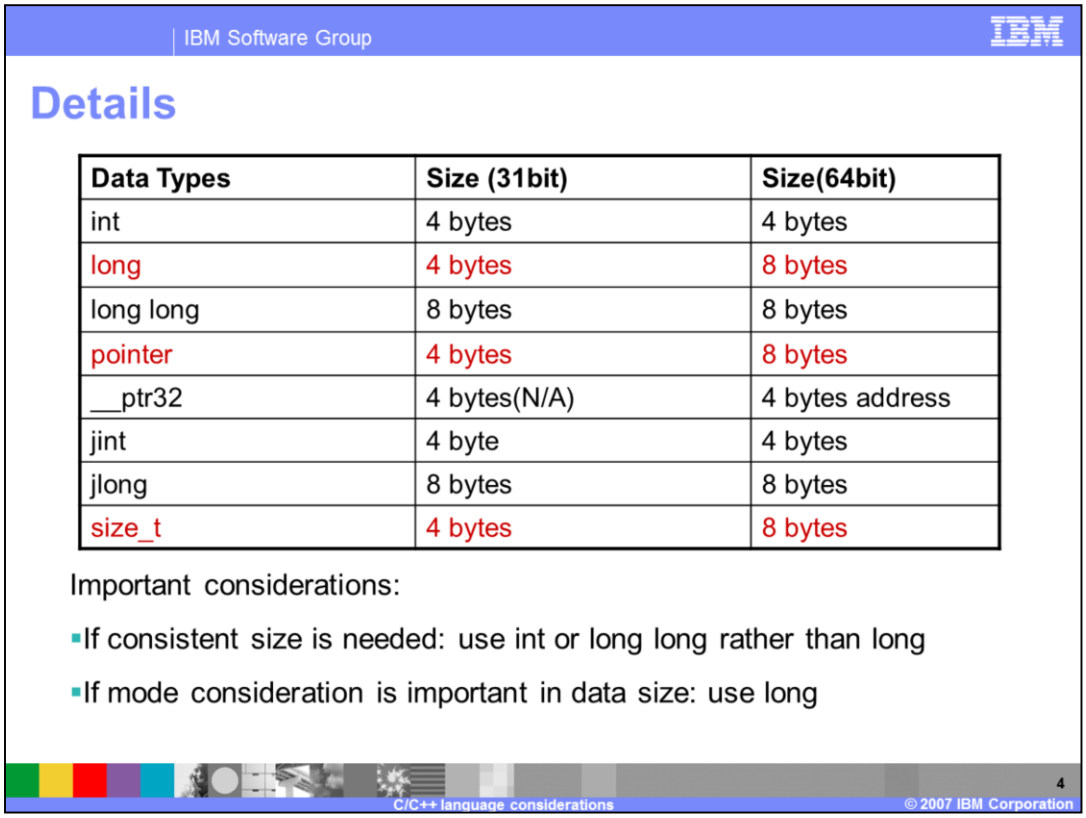
This presentation covers the C/C++ language considerations when using 64-bit support in a WebSphere® Base Application Server V6.1 on z/OS.



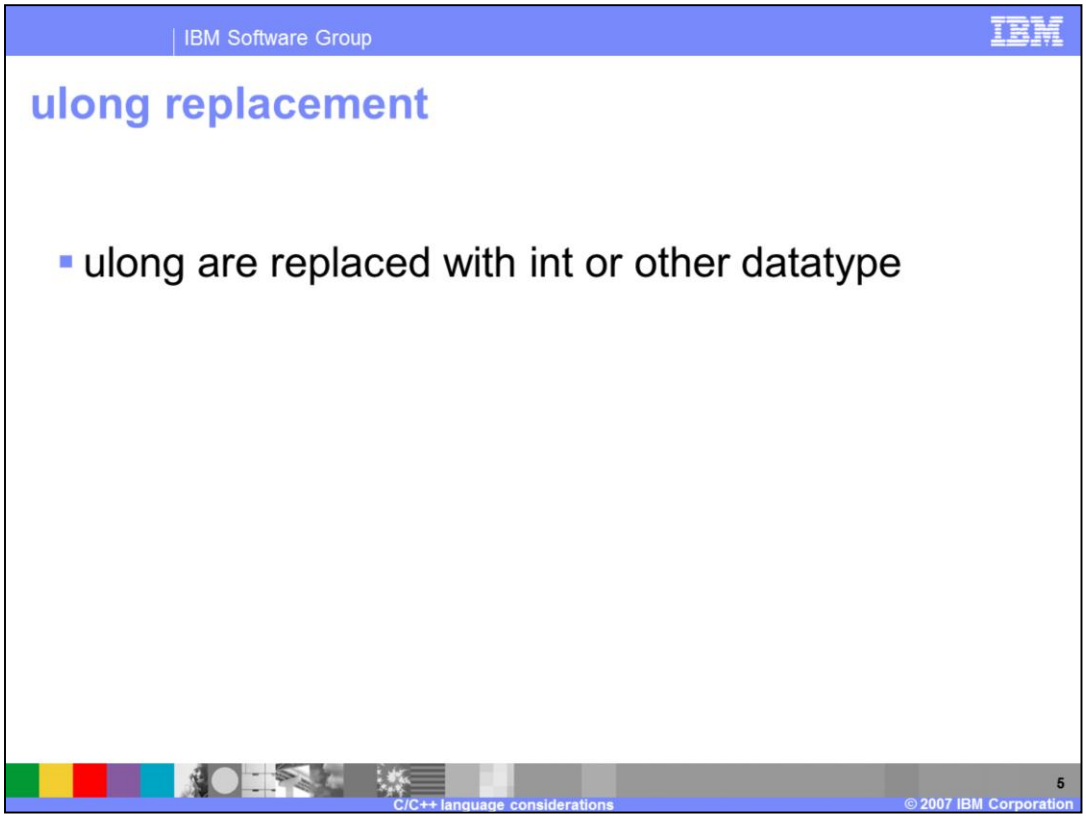
The agenda includes a discussion of data types and macros that were changed or added to enable consistency between 31-bit and 64-bit modes.



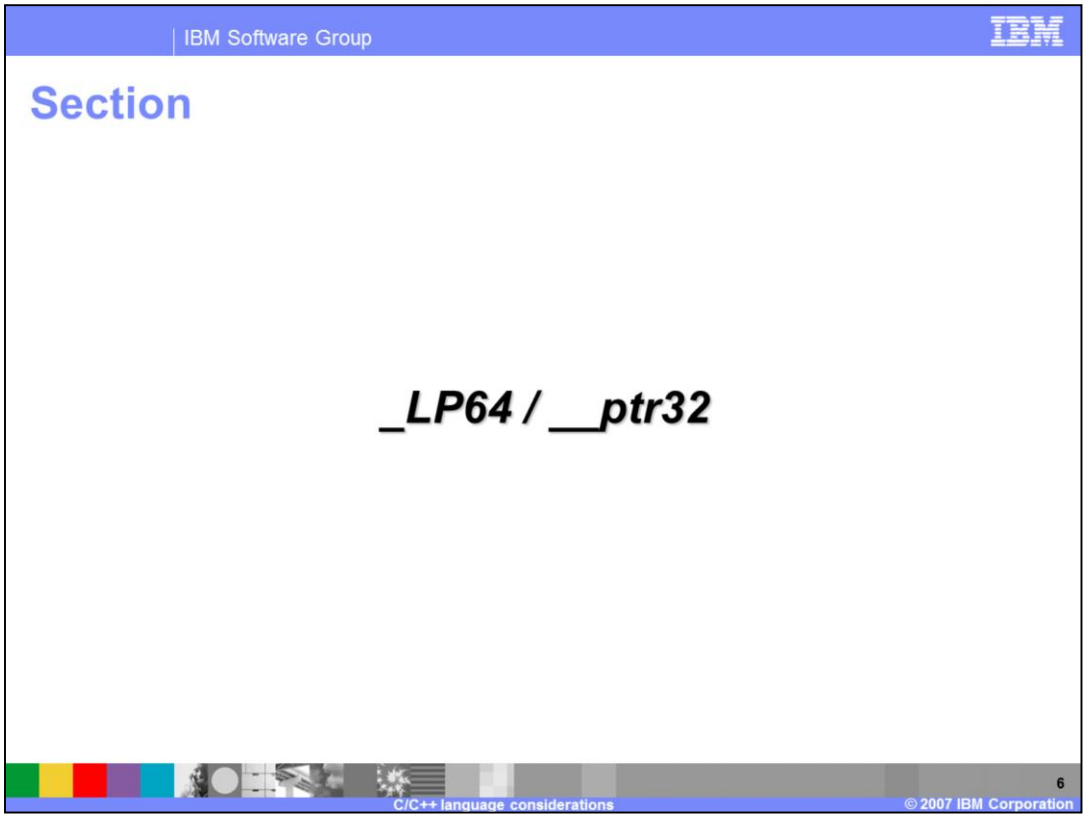
This section covers data type differences between 31-bit and 64-bit modes.



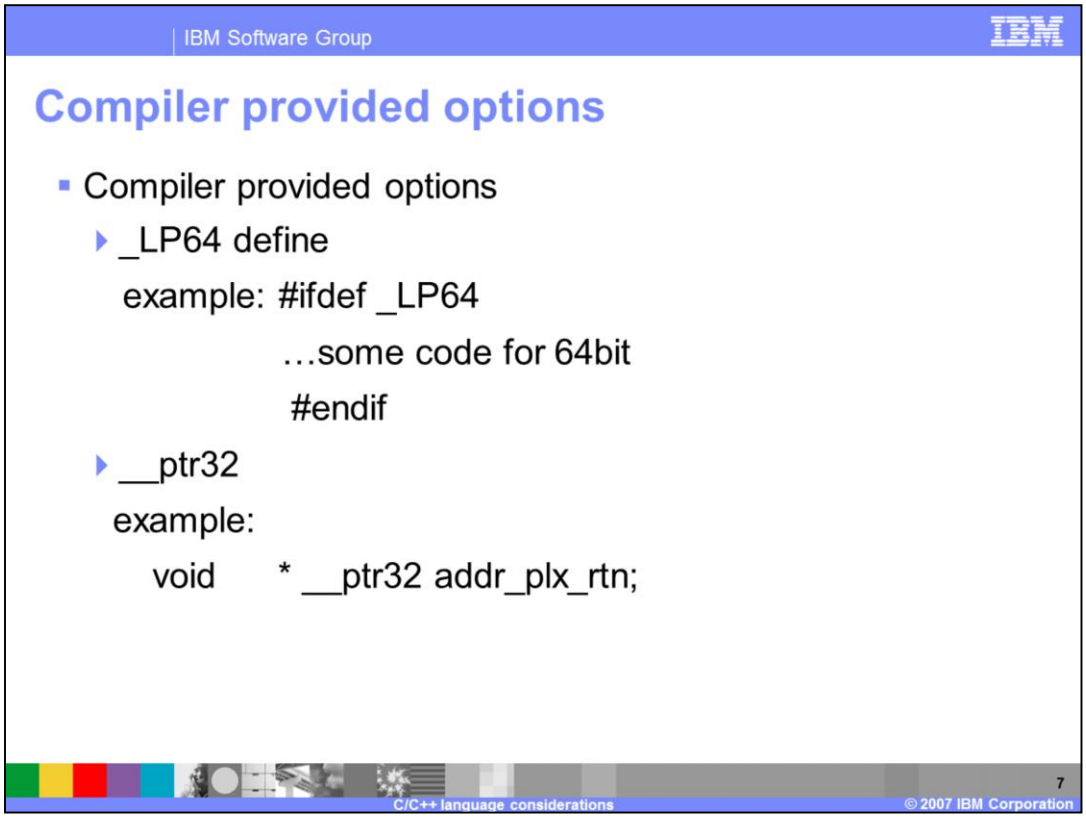
Depending on which mode code is compiled for, some C/C++ native code datatype sizes are changed. The table lists datatype, mode and defined size. These data types should be understood and implemented accordingly.



The ulong datatype size changes in C/C++ native code based on either 31-bit or 64-bit mode. From a common code point of view, this change breaks the structure mapping between C/C++ and PLX. To solve this problem, the ulong type is converted to an int type, which is consistent between 31-bit and 64-bit mode. Other datatypes can be used based on structure requirements.

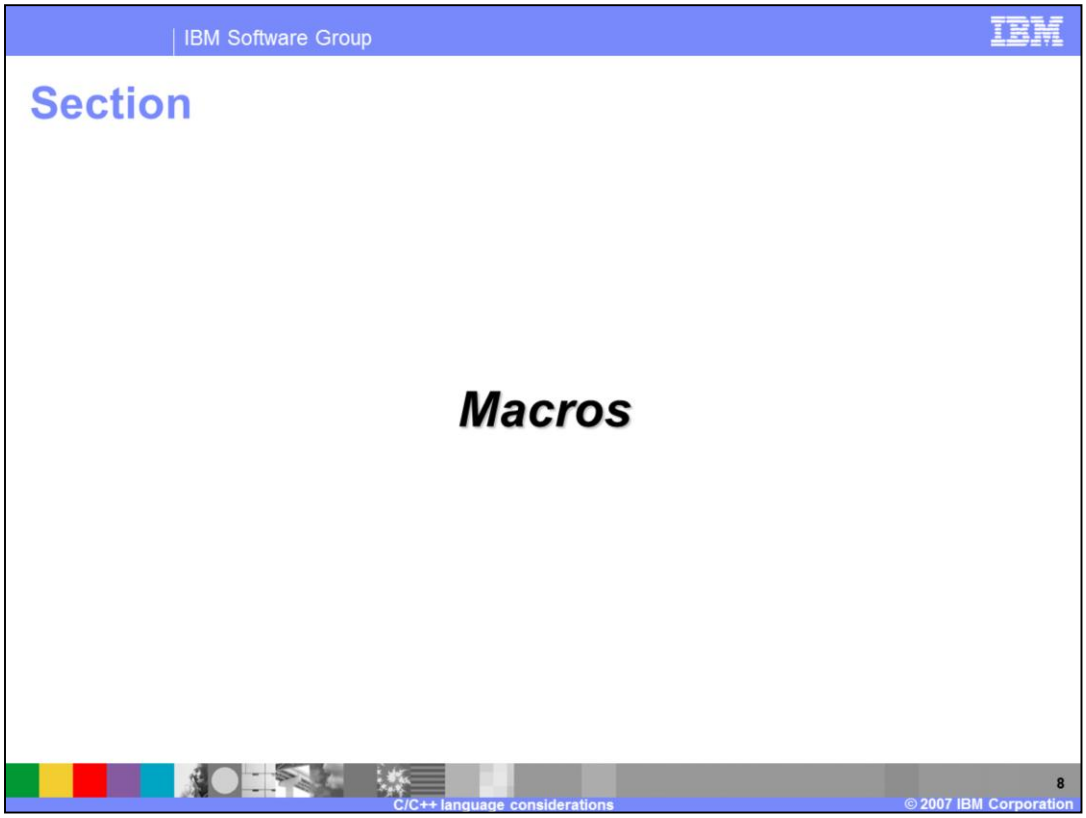


This section covers _LP64 and __ptr32.

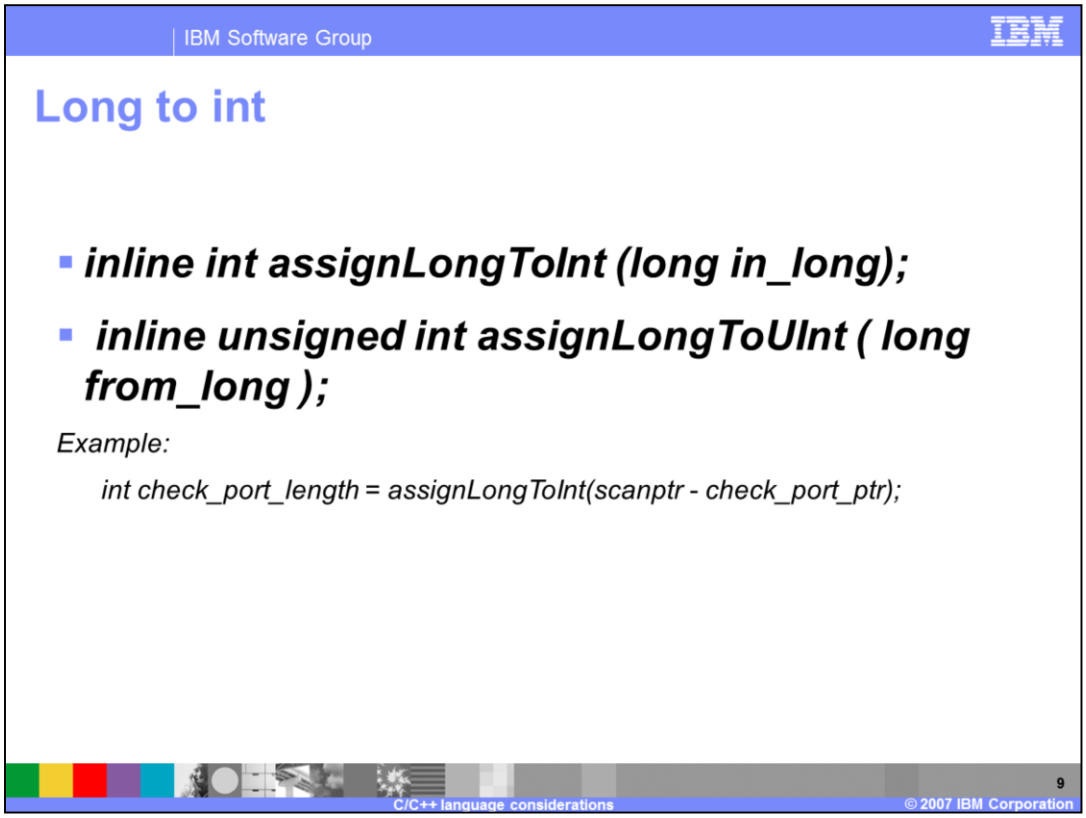


_LP64 is a compiler-provided directive that allows 64-bit code to be separated from 31-bit code in common source code through the use of #ifdef.

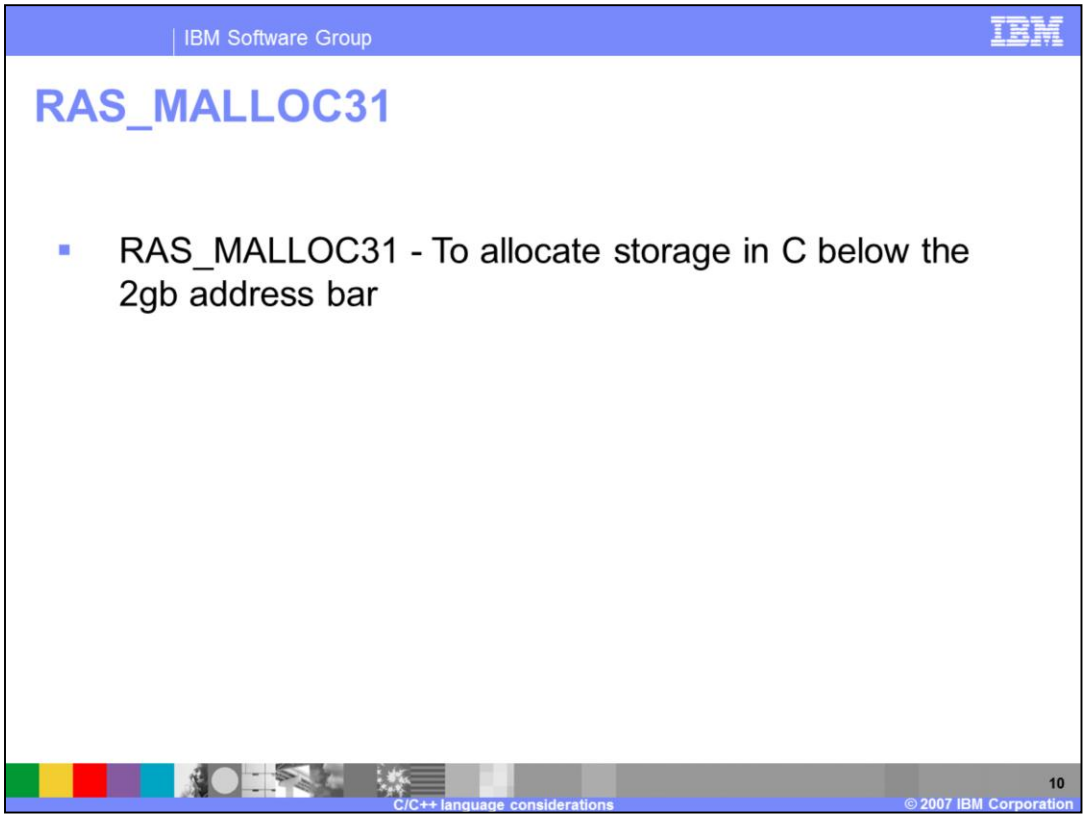
The __ptr32 attribute is used to make the pointer datatype 4 bytes under 64-bit compile mode. This is useful in cases where storage is obtained under the 2 gigabyte address bar and the address must be stored in the 64-bit runtime for PLX to manipulate later. This option is ignored in 31-bit compile mode.



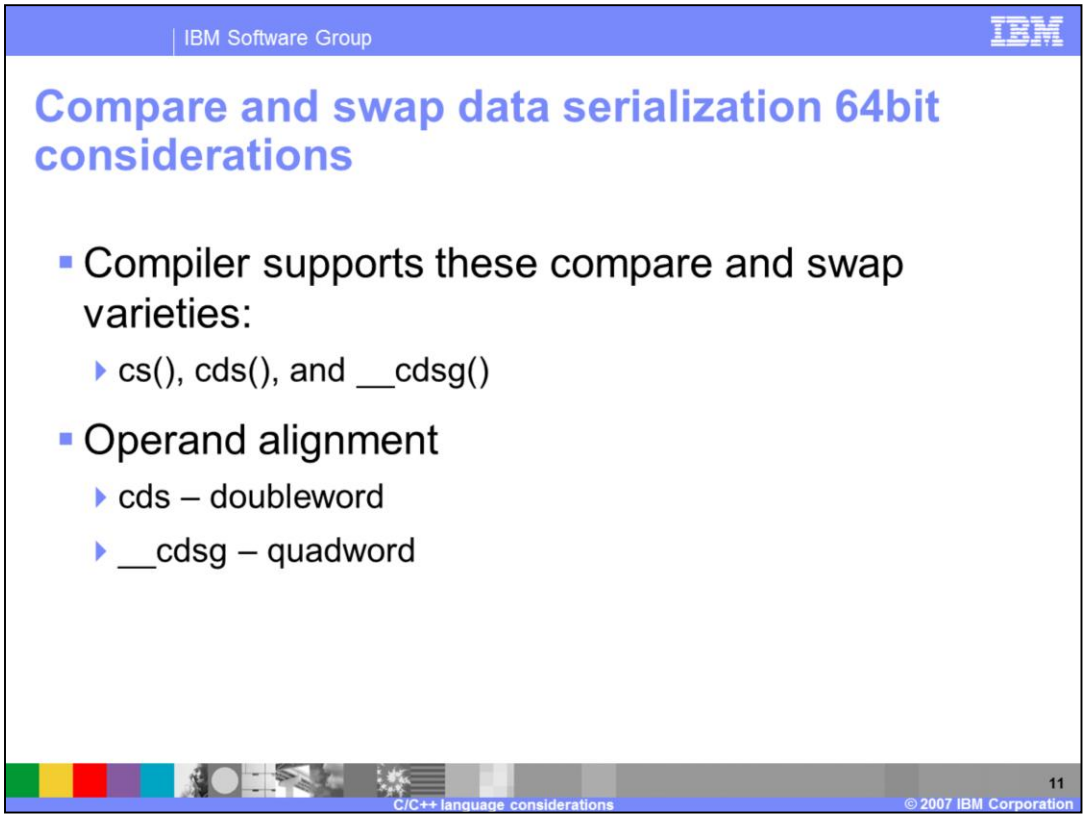
This section covers macros.



Pointer arithmetic is required in native code to calculate length or size. When the calculated value fits in a 4 byte data type, which is true in most cases, the data can be stored as an int. However, 64-bit pointers are 8 bytes and normally the difference of two pointers can not be stored into an int directly due to a compiler limitation. For this reason, the above macros are added to ensure 31-bit / 64-bit code consistency. If the difference between the two pointer values is too big to fit in a 4byte field or the value is negative an exception is thrown.

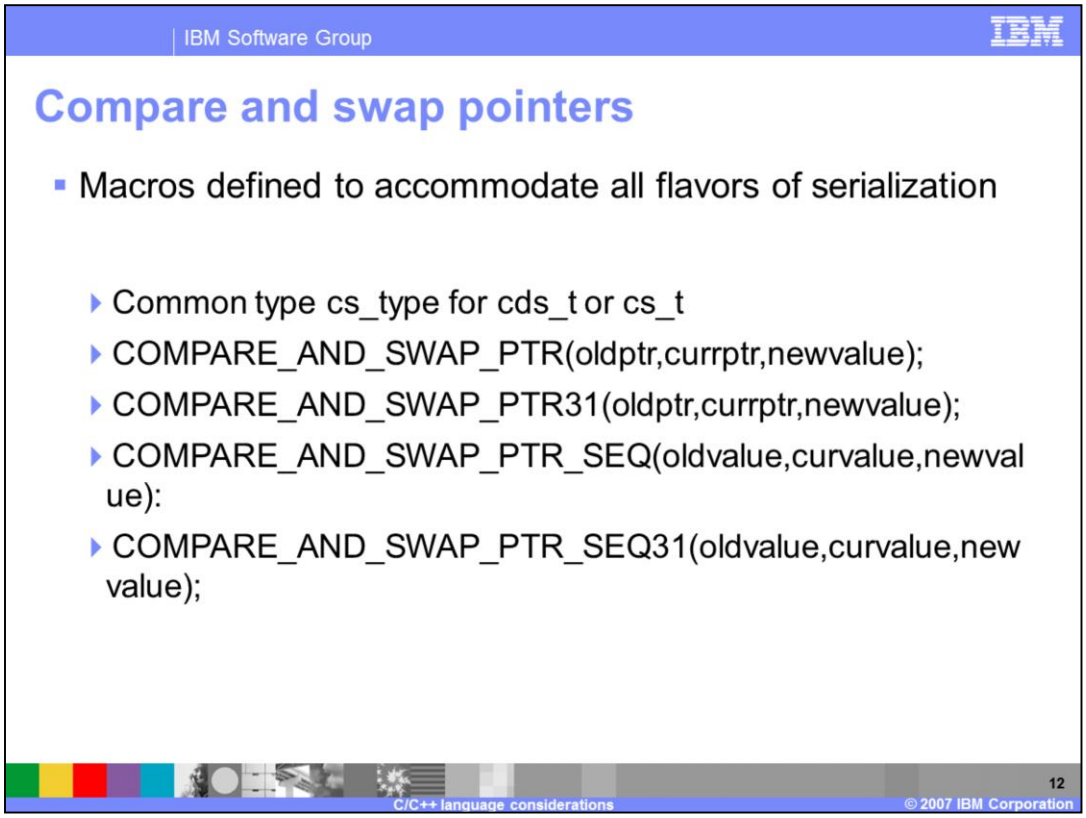


PLX code runs in 31-bit mode under a 64-bit WebSphere Application Server V6.1 runtime. To pass data between 64-bit C/C++ storage to PLX, storage must be allocated in C/C++ below the 2 gigabyte address bar so PLX can manipulate them and this macro is provided for this purpose.



In general, the compiler supports the compare and swap commands (shown above) for 4, 8, and 16 byte operands. Cds() and __cdsg() require that the operand be double word and quad word aligned.

To keep native code common between 31-bit and 64-bit processing, a variety of macros are provided to keep these operations consistent.



The macros are as follows:

1. COMPARE_AND_SWAP_PTR (old pointer, current pointer, new value) is provided to comply with compile mode for 4 or 8-byte pointers.

2. COMPARE_AND_SWAP_PTR31(old pointer, current pointer, new value) is used on 4 byte addresses regardless of bit mode

3. COMPARE_AND_SWAP_PTR_SEQ(old value, current value, new value) is used where an 8-byte pointer and 8-byte sequence number is used

4. COMPARE_AND_SWAP_PTR_SEQ31(old value, current value, new value) is used where a 4-byte pointer and a 4-byte sequence number is used regardless of mode.

cs_type is a common type created for cds_t or cs_t so code can expand based on compile mode.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere z/OS

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance 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 or performance improvements equivalent to the ratios stated here.